НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

«На правах рукопису»

УДК _____ ___

«До захисту допущено»

Завідувач кафедри

_____ *С. Г. Стіренко*

(підпис) (ініціали, прізвище)

“ ” _____ 20***20*** р.

Магістерська дисертація

зі спеціальності: *121. Інженерія програмного забезпечення*

(код та назва напряму підготовки або спеціальності)

Спеціалізація: *121. Інженерія програмного забезпечення комп'ютерних систем*

на тему: Багатомодульна клієнтська платформа трейдингу транспортними засобами

Виконав: студент 6 курсу, групи ІП-з92мп

(шифр групи)

Цехмейструк Роман Вікторович

(прізвище, ім'я, по батькові) (підпис)

Науковий керівник проф., д.т.н., проф., Жабін В. І.

(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____

(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____

(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

(підпис)

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *Обчислювальної техніки*
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність *121. Інженерія програмного забезпечення*
(код і назва)

Спеціалізація *121. Інженерія програмного забезпечення комп'ютерних систем*
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
С. Г. Стіренко
(підпис) (ініціали, прізвище)
« » 20***20*** р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Цехмейструку Роману Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації *Багатомодульна клієнтська платформа для трейдингу транспортними засобами*
Науковий керівник дисертації *Жабін Валерій Іванович, д.т.н., проф.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «26» *жовтня 2020* р. № *3133-с*
2. Строк подання студентом дисертації *25 листопада 2020*
3. Об'єкт дослідження *процес побудови багатомодульної архітектури для реалізації платформи для трейдингу транспортними засобами*
4. Предмет дослідження *метод поєднання окремих незалежних модулів в цілісний готовий продукт*
5. Перелік завдань, які потрібно розробити: *Опис предметної області, дослідження існуючих систем, побудова архітектури, реалізація власного додатку для системи Android, який буде спілкуватися з сервером через REST Api.*

6. Консультанти розділів дисертації:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Розділ 1. Огляд існуючих рішень			
Розділ 2. Вибір засобів розробки і			
Розділ 3. Реалізація додатку			
Розділ 4. Інструкція користувача і			
Розділ 5. Розробка стартап-проекту			

7. Дата видачі завдання _____

Календарний план

№ п/п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів роекту(роботи)	Примітки
1..	*Затвердження теми роботи*	*18.12.2019*	
2.	*Вивчення та аналіз завдання*	*20.05.2020-02.06.2020*	
3.	*Розробка загальної архітектури та вибір технологій проекту*	*03.06.2020-20.07.2020*	
4.	*Розробка структур окремих підсистем*	*21.07.2020-29.07.2020*	
5.	*Програмна реалізація системи*	*30.07.2020-30.08.2020*	
6.	*Оформлення пояснювальної записки*	*01.09.2020-02.11.2020*	
8.	*Передзахист*	*26.11.2020*	
9.	*Захист*	*18.12.2020*	

Студент _____ _____
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ _____
(підпис) (ініціали, прізвище)

1. НАЙМЕНУВАННЯ І ГАЛУЗЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку багатомодульної клієнтської платформи для трейдингу транспортними засобами.

Область використання: процеси купівлі та продажу майна, налаштування рішення для бізнесу.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут» (НТУУ «КПІ» ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для надання можливості пошуку вживаних та нових автомобілів з використанням інтернет-технологій з клієнтської частини, яка буде працювати на Android-пристроях.

4. ДЖЕРЕЛО РОЗРОБКИ

Джерелом розробки є технічне завдання, згідно з яким потрібно реалізувати клієнтську платформу для трейдингу транспортними засобами з можливістю перевикористання окремих модулей в інших проектах.

5. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

5.1. Вимоги до розроблюваного продукту

Додаток повинен забезпечити такі основні функції:

- Повинен надавати можливість створювати обліковий запис користувача в системі та входу в систему.
- Повинен надавати можливість пошуку всіх доступних автомобілів за обраними фільтрами.
- Повинен надавати можливість модератору сервісу перевіряти нові додані оголошення перед тим, як вони потраплять у список всіх доступних оголошень.
- Повинен надавати можливість завантажувати своє оголошення на дошку.
- Повинен надавати можливість зв’язатися з автором оголошення.

Розробку системи виконати використовуючи мову програмування Kotlin, з підключенням фреймворку Kotlin Coroutines для забезпечення обробки багатопотоковості та фреймворку Dagger 2 для досягнення цілей впровадження залежностей. Архітектура повинна бути мультимодульною, кожен модуль повинен бути незалежним і самодостатнім з можливістю перевикористання його в інших проектах.

5.2. Вимоги до програмного забезпечення

Програмне забезпечення повинно відповідати таким вимогам:

- швидкість роботи та конфігурації даних користувача;
- простота експлуатації;
- надійність.

6. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

- Титульний лист.
- Лист завдання.
- Анотації.
- Опис альбому.

- Технічне завдання.
- Зміст пояснювальної записки.
- Перелік умовних позначень.
- Пояснювальна записка.
- Висновки.
- Список літератури.
- Лістинг програми.

7. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою роботи	20.03.2020
Розроблення та узгодження технічного завдання	30.03.2020
Розроблення структури програмного додатку	03.04.2020
Розроблення графічного інтерфейсу користувача	18.04.2020
Програмна реалізація бізнес-логіки додатку	01.09.2020
Програмна реалізація графічного інтерфейсу користувача	14.09.2020
Тестування та налагодження додатку	21.09.2020
Підготовка матеріалів текстової частини проекту	20.10.2020

РЕФЕРАТ

на магістерську дисертацію

виконану на тему: Багатомодульна клієнтська платформа для трейдингу транспортними засобами

студентом: Цехмейструком Романом Вікторовичем

Робота складається із вступу та п’ятьох розділів. Загальний обсяг роботи: 74 аркуші основного тексту, 22 ілюстрації, 24 таблиць. При підготовці використовувалася література з 14 різних джерел.

Актуальність. Процес продажу або покупки уживаних машин все ще будується переважно на приватній основі, а тому вкрай хаотичний і криміналізований. Аналіз виявив тенденції, які забезпечують стрімкий розвиток вторинного ринку автомобілів, стимульований фінансовою кризою. Ринок стає організованим, підвищується рівень обслуговування. На вторинному авторинку торгівля автомобілями стала окремим бізнесом. Полягаючись на описане вище, бачимо, що актуальність створення платформи для торгівлі транспортними засобами стає все більш високою.

Мета і завдання дослідження. Метою магістерської роботи є покращення підходів до розробки програмних продуктів використовуючи багатомодульний підхід, а також створення платформи для надання послуг з продажу та купівля нових або вживаних транспортних засобів з використанням можливості віддаленої конфігурації продукту.

Об’єкт дослідження – процес побудови багатомодульної архітектури для реалізації платформи для трейдингу транспортними засобами.

Предмет дослідження – метод поєднання окремих незалежних модулів в цілісний готовий продукт.

Методи досліджень. Для досягнення поставлених в магістерській роботі задач використано методи багатомодульної побудови архітектури програмних додатків.

Наукова новизна одержаних результатів роботи полягає у наступному:

- Побудований механізм взаємодії різних модулів, функціонал яких знаходиться в межах 1 екрану користувача.

- розроблено програмний продукт для надання послуг купівлі та продажу авто з використанням механізму, описаного вище.

Проведене дослідження дає можливість створення програмних продуктів таким чином, що кожна частина, яка відповідає за певний функціонал, може бути перевикористаною в інших проектах незалежно від їх технологій або архітектури.

Особистий внесок здобувача. Магістерське дослідження є самостійно виконаною роботою, в якій відображено особистий авторський підхід та особисто отримані теоретичні та прикладні результати, що відносяться до проектування архітектури програмного коду додатків, які працюють на платформі Android. Формулювання мети та завдань дослідження проводилось спільно з науковим керівником.

Практична цінність. Отримані результати можуть використовуватися у майбутніх дослідженнях за напрямками:

- збільшення швидкості роботи незалежних команд в результаті винесення функціоналу в окремі незалежні модулі;
- прогнозування кількості ресурсів, необхідних на розробку функціоналу.

Конференції:

Ключові слова

Багатомодульна архітектура, MVVM, трейдинг транспортними засобами, розробка продукту незалежними командами.

ABSTRACT

for a master's thesis

made on the topic: Multimodular client platform for vehicle trading

student: Tsekhmeistruk Roman Viktorovich

The work consists of an introduction and five sections. Total volume of work: 94 sheets of the main text, 22 illustrations, 27 tables. Literature from 8 different sources was used in the preparation.

Topicality. The process of selling or buying used cars is still built mainly on a private basis, and therefore extremely chaotic and criminalized. The analysis revealed trends that ensure the rapid development of the secondary car market, stimulated by the financial crisis. The market becomes organized, the level of service increases. In the secondary car market, car trade has become a separate business. The boom began, and many companies, both official dealers and independent traders with a base for pre-sales training or car service partnerships, started selling used cars.

Based on the above, we see that the relevance of creating a platform for trade in vehicles is becoming increasingly high.

The purpose and objectives of the study. The purpose of the master's thesis is to create a platform for the provision of services for the sale and purchase of new or used vehicles using the possibility of remote product configuration.

The product must be implemented in such a way that each of its modules can be reused for other similar products.

The object of research – is the use of a multi-module architecture in the construction of an application for the sale of vehicles.

The subject of research – is the interaction of modules in multimodule platforms.

Research methods. To achieve the objectives set in the master's thesis, methods of multimodule construction of software application architecture are used.

The scientific novelty of the obtained results is as follows:

• The mechanism of interaction of various modules which functionality is within 1 user screen is constructed.

• a software product has been developed to provide car buying and selling services using the mechanism described above.

The study makes it possible to create software products in such a way that each part that is responsible for a particular functionality can be reused in other projects, regardless of their technology or architecture.

Personal contribution of the applicant. The master's research is a self-performed work, which reflects the personal author's approach and personally obtained theoretical and applied results related to the design of the software code architecture of applications running on the Android platform. The formulation of the purpose and objectives of the study was carried out jointly with the supervisor.

Practical value. The obtained results can be used in future research in the following areas:

• increase in speed of work of independent commands as a result of removal of functionality in separate independent modules;

• forecasting the amount of resources needed to develop functionality.

Conferences:

Keywords

Multimodular architecture, MVVM, vehicle trading, product development by independent teams.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до магістерської дисертації

на тему: Багатомодульна клієнтська платформа для трейдингу транспортними засобами

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

DI (Dependency Injection) Впровадження залежностей.

GUI (Graphical User Interface) Графічний інтерфейс користувача.

MVP (Model View Presenter) Патерн розробки коду.

MVVM (Model View ViewModel) Патерн розробки коду

API (Application Programming Interface) Інтерфейс програмування застосунків.

JSON (JavaScript Object Notation) текстовий формат обміну даними.

XML (eXtensible Markup Language) Розширювана мова розмітки.

CI (Continuous Integration) Неперервна інтеграція.

REST (Representational state transfer) Підхід по архітектури мережевих протоколів.

SDK (Software Development Kit) Набір засобів розробки.

ВСТУП

Процес продажу або покупки уживаних машин все ще будується переважно на приватній основі, а тому вкрай хаотичний і криміналізований. Аналіз виявив такі тенденції:

• наявність високого відкладеного попиту на автомобілі з пробігом. Багато хто хотів би купити іномарки, але для основної маси населення автомобілі ряду брендів недоступні за ціною. Внаслідок цього відкладений попит наростає;

• п'ята частина власників хочуть замінити новий автомобіль в першому ж році експлуатації, чверть - через два роки і третина - через три, як це робиться на західних ринках;

• пропозиції старих автомобілів наростають такими темпами, що продажі громадянами один одному скоро стануть просто неможливими - в хаосі приватного ринку продавці та покупці заблукають і віддадуть перевагу звертатися в дилерські центри і незалежні торговельні фірми;

• покупцям необхідний цивілізований ринок автомобілів з пробігом, щоб фінансові ризики придбання колишнього в експлуатації автомобіля були мінімальні, а гарний технічний стан транспортного засобу гарантовано. Власники великих парків автомобілів (прокатні та лізингові компанії, транспортні підприємства, великі корпорації) неминуче стикаються з проблемою реалізації автомобілів, що вийшли з експлуатації.

Ці тенденції забезпечують стрімкий розвиток вторинного ринку автомобілів, стимульований фінансовою кризою. Ринок стає організованим, підвищується рівень обслуговування. На вторинному авторинку торгівля автомобілями стала окремим бізнесом. Почався бум, і продажем автомобілів, що були в експлуатації, зайнялися багато фірм - як офіційні дилери, так і незалежні фірми-трейдери, які мають базою для передпродажної підготовки або партнерськими зв'язками з автосервісу.

Полягаючись на описане вище, бачимо, що актуальність створення платформи для торгівлі транспортними засобами стає все більш високою.

РОЗДІЛ 1

ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1. Загальні відомості про платформу для трейндингу транспортними засобами

Майже кожен власник автомобілем стикається в житті з продажем авто. Власник автомобіля продає її з різних причин: авто не підлягає ремонту, змінилися смаки і переваги, автомобіль капітально застарів або вирішив замінити на новий. У зв'язку з цим з'являються проблеми з продажем автомобіля. Хочеться швидше вирішити проблему, а також отримати величезну кількість грошей. Ось тому ми поділимося з вами рекомендаціями і порадами, як спритно і прибутково продати своє авто.

Специфіка торгівлі автомобілями полягає в тому, що постачальники, будучи джерелом цього товару, не створюють торгово-сервісних мереж. Замість цього вони пропонують автомобілі з пробігом всім бажаючим, не беручи на себе відповідальності за подальше обслуговування і забезпечення запасними частинами, як це прийнято в торгівлі новими машинами. Більш того, не постачальники вибирають трейдерів, а трейдери вибирають постачальників, погодившись із своїми ремонтними потужностями і можливостями збуту.

Необхідно провести аналіз існуючих рішень, доступних на ринку.

1.2. Порівняння існуючих рішень

1.2.1. Autolist

Autolist - це програма для вживаних автомобілів та вантажівок, подібна до таких програм, як AutoTrader. У ній є функції, такі як пошук, сортування та збереження вибраного. У ньому також є попередження про зниження ціни та доступні десятки тисяч транспортних засобів. Ви також можете переглянути історію цін на транспортні засоби, які наявні в продажі. Це також джерела з багатьох інших веб-сайтів. Це робить програму пристойним всебічним

рішенням для пошуку покупців автомобілів. Інтерфейс базовий, але ефективний, і виглядає непогано.

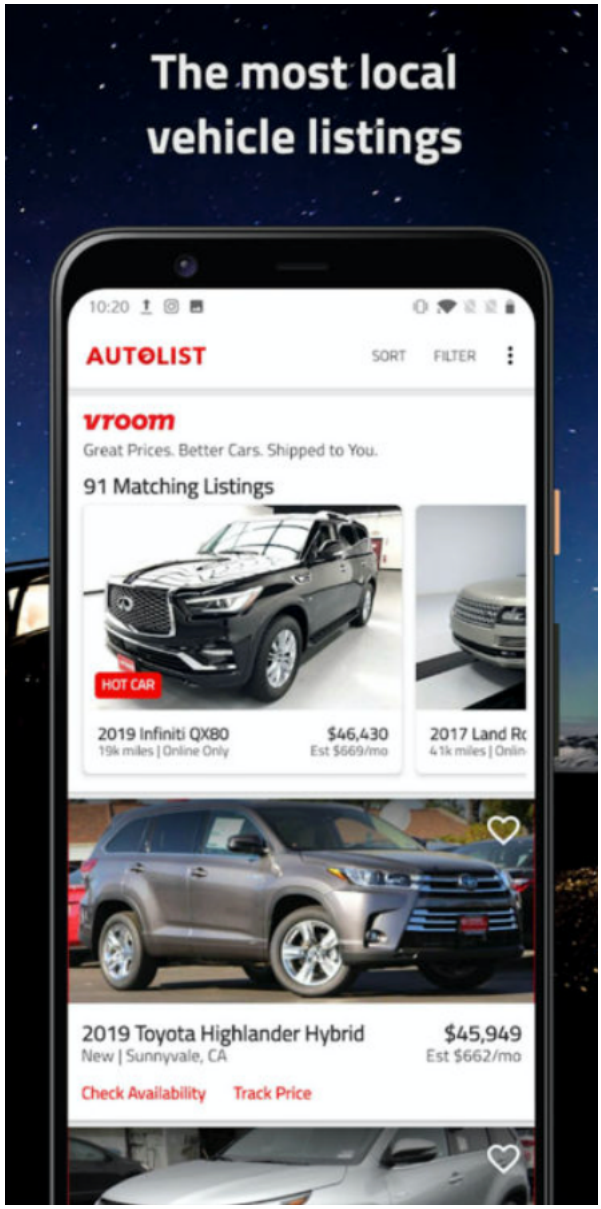

Рис. 1.1. Візуальний вигляд сторінки програму Autolist [5]

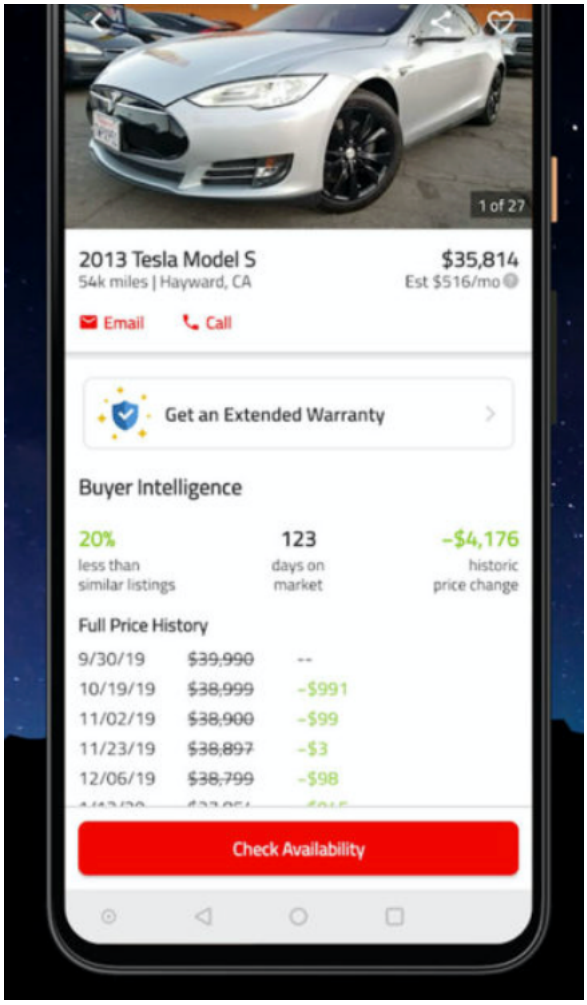

Рис. 1.2. Візуальний вигляд сторінки деталей оголошення додатку Autolist [8]

1.2.2. AutoScout24

AutoScout24 - одна з небагатьох хороших програм для покупок автомобілів у Європі. Він може похвалитися індексом понад два мільйони автомобілів по всьому континенту. Ви можете проводити місцеві пошуки, зберігати вибране для перегляду в автономному режимі та зв’язуватися з продавцем через додаток. Він також має опцію сортування для таких речей, як

ціна, відстань, пробіг, дата реєстрації тощо. Додаток також дає вам можливість розмістити свою машину.

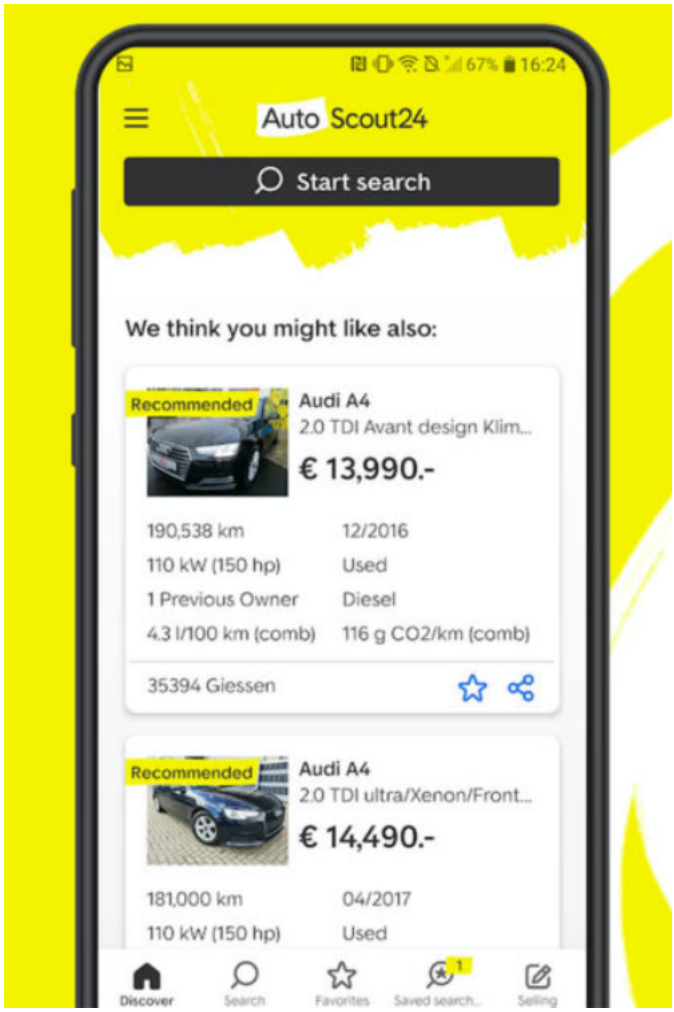

Рис. 1.3. Візуальний вигляд програми AutoScout24 [5]

1.2.3. AutoTrader

AutoTrader - одна з найкращих програм для покупок автомобілів у США. Вона дозволяє шукати нові та вживані машини поблизу вашого місцезнаходження. Крім того, він має набір фільтрів, завдяки чому ви можете знайти саме ту машину, яку шукаєте. Фільтри не завжди працюють, але є корисними. Ви також можете зберегти вибране для подальшого перегляду, зв’язатися з власником через додаток. Додаток можна безкоштовно переглядати, але продати свою машину є платною опцією.

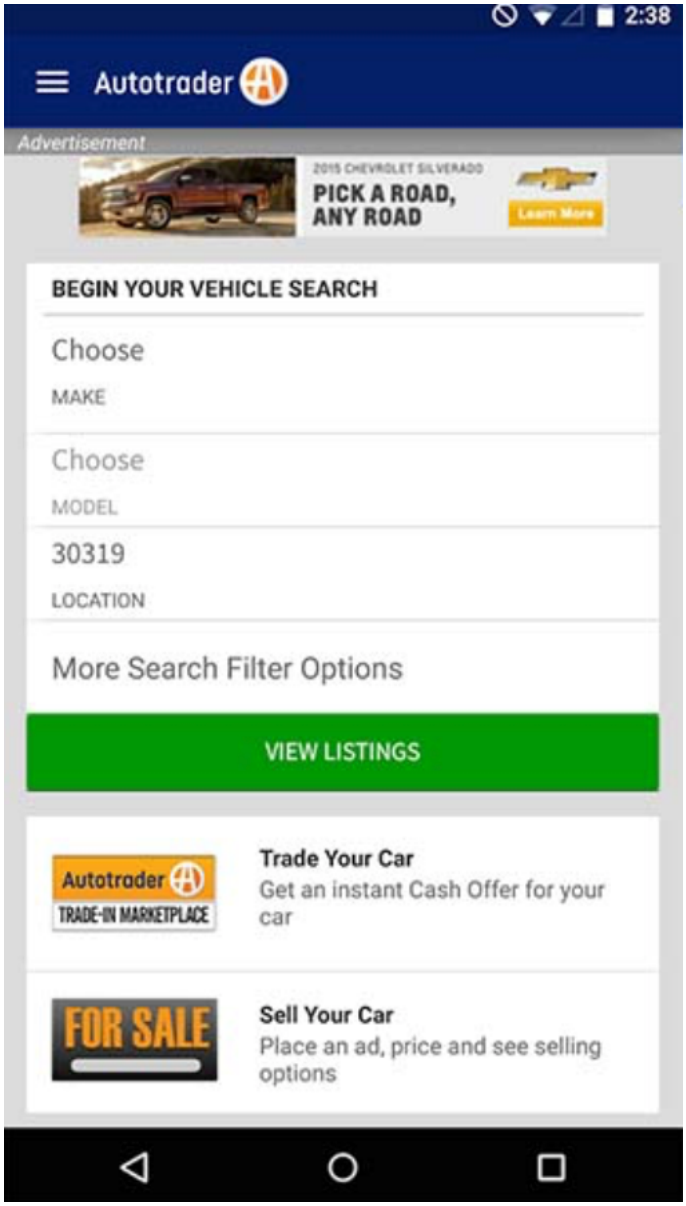

Рис. 1.4. Візуальний вигляд стартової сторінки AutoTrader [9]

1.2.4. CarFAX Used Cars

CARFAX - дуже корисна програма для тих, хто купує автомобіль. Розробники можуть похвалитися мільйонними запасами. Люди можуть використовувати функцію сортування, щоб знайти потрібні транспортні засоби. До кожного авто додається звіт CARFAX, щоб ви могли побачити, що з авто насправді не так. Також є функція збереження вибраних для подальшого перегляду. Також є можливість переглянути звіти CARFAX щодо автомобілів,

не вказаних у додатку. Це буде коштувати вам грошей. Однак це має сенс, оскільки саме так CARFAX використовує для заробітку.

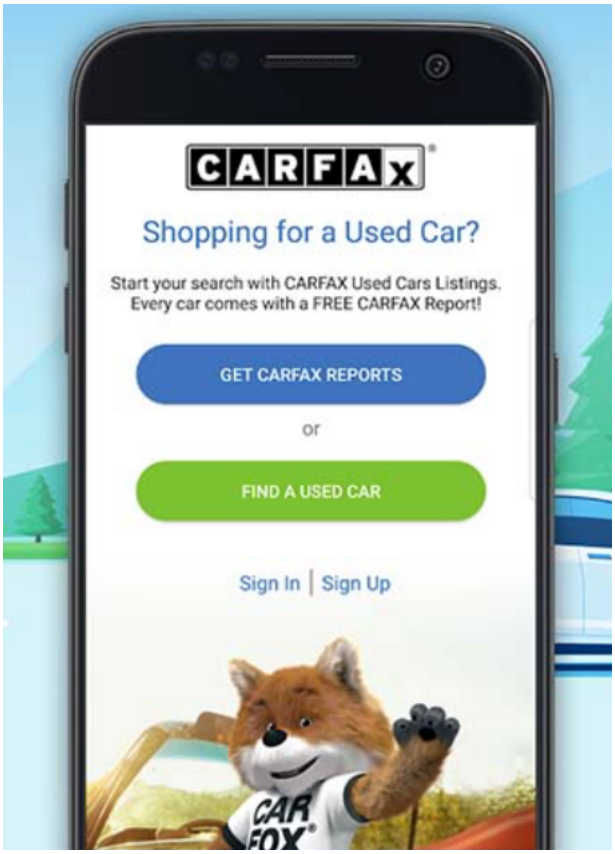

Рис. 1.5. Візуальний вигляд екрану старту додатку CarFax Used Cars [11]

1.2.5. CarMax

CarMax - одна з найкращих програм для покупок автомобілів у США. Ви можете шукати машини на складі. Крім того, ви можете зберігати обрані, попереджувати про такі речі, як зниження цін та нові збіги для нових автомобілів, додані в інвентар. Існує навіть калькулятор платежів для тих, хто хоче це зрозуміти. Інтерфейс чистий і простий у використанні. Ви також

можете використовувати додаток для здійснення платежів до CarMax після покупки автомобіля.

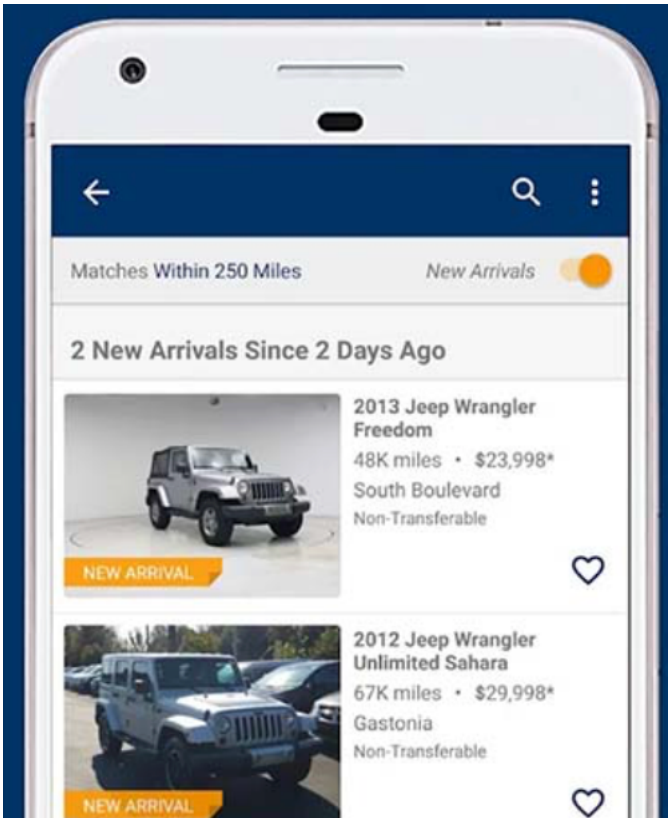

Рис. 1.6. Візуальний вигляд меню додатку CarMax [11]

1.2.6. CarMudi

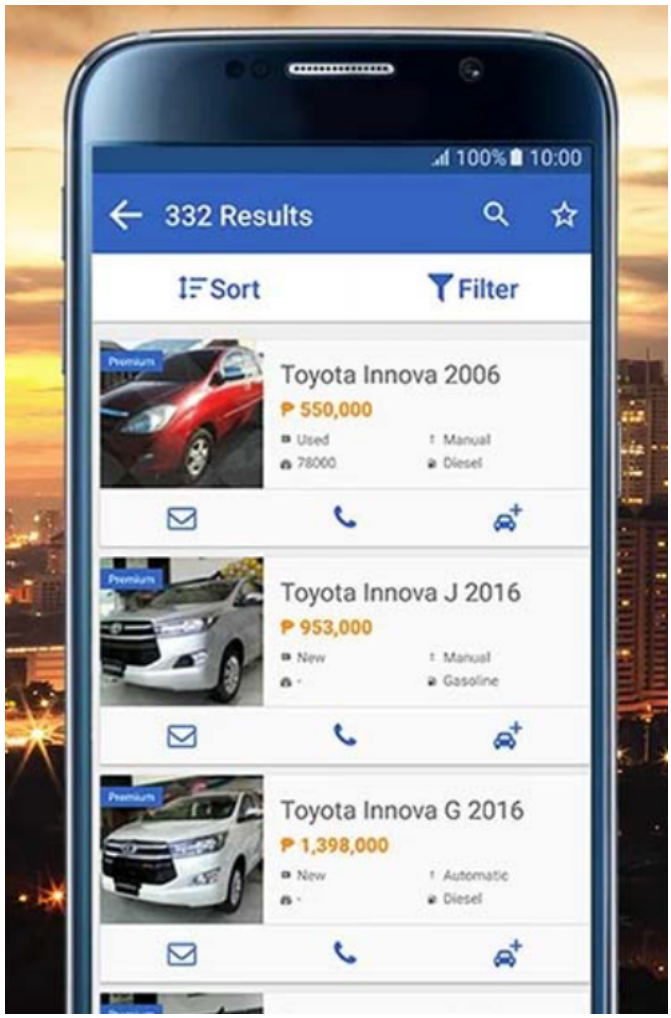

Рис. 1.7. Візуальний вигляд стартового екрану додатку CarMudi [9]

Carmudi - це широкодоступний додаток для покупок автомобілів. Він підтримує сім країн Північної Америки, Азії та Близького Сходу. Він також може похвалитися понад 200 000 автомобілів на продаж. Є функції як для покупців, так і для продавців. Покупці можуть зберігати обране, шукати з сортуванням та переглядати покупців на основі відгуків. Продавці можуть швидко заповнити дані про свій автомобіль та поговорити з потенційними покупцями. Перелік підтримуваних країн можна знайти на сторінці Google Play Store.

1.2.7. CarVanna

Carvana - цікавий варіант придбання автомобіля. Це інтернет-ринок, який приймає замовлення вашого автомобіля і фактично доставляє автомобіль

прямо до вашого будинку. Послуга сприяє безконтактному досвіду доставки, і ви можете придбати будь-яку машину, яку хочете, за умови отримання фінансування або наявності грошей на це. Послуга також забезпечує фінансування. Він може похвалитися колекцією з понад 20 000 автомобілів, і є семиденна гарантія повернення грошей, якщо вам це не подобається. Я насправді бачив, як ця служба доставляє автомобіль у реальному житті, тому це законно.

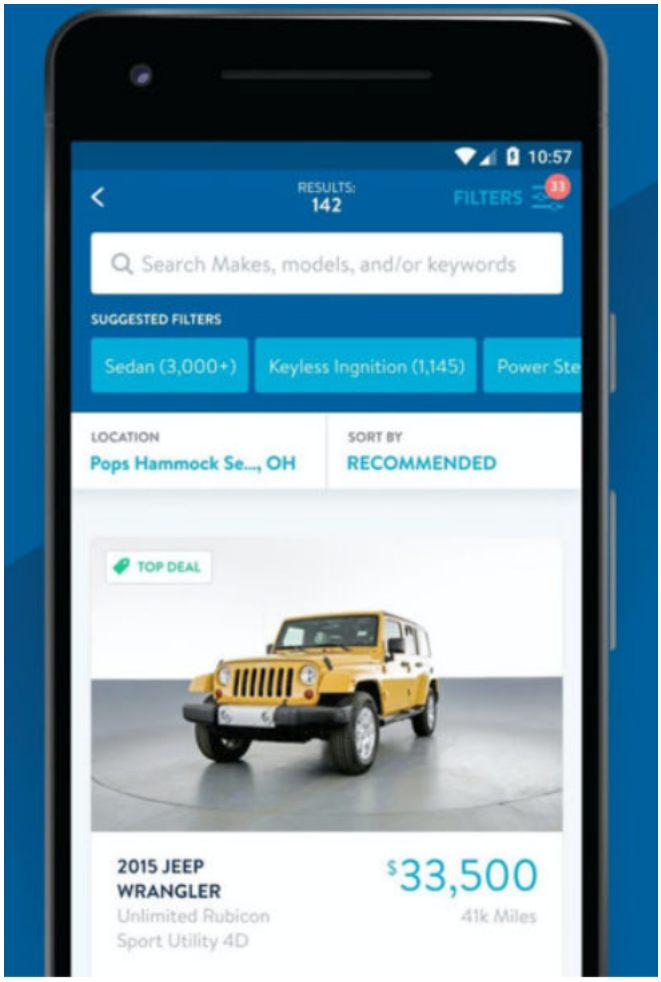

Рис. 1.8. Візуальний вигляд стартового екрану додатку CarVanna [8]

1.2.8. TrueCar

TrueCar - ще один величезний ресурс для покупців автомобілів. Він може похвалитися запасами понад 13000 дилерів по всій території США. Вони також можуть похвалитися запасом понад мільйон автомобілів. Ви можете шукати як нові, так і вживані машини з різних дилерських центрів. Крім того,

він має допоміжний персонал, який допоможе вам у процесі. У них є окремий додаток для дилерів та продавців. Деякі інші функції включають ціновий путівник, який показує, скільки інші люди платять за транспортний засіб, пристойний і простий інтерфейс користувача та деякі інші інструменти для покупки автомобілів. Це набагато простіше, ніж переходити з місця дилера на сайт дилера.

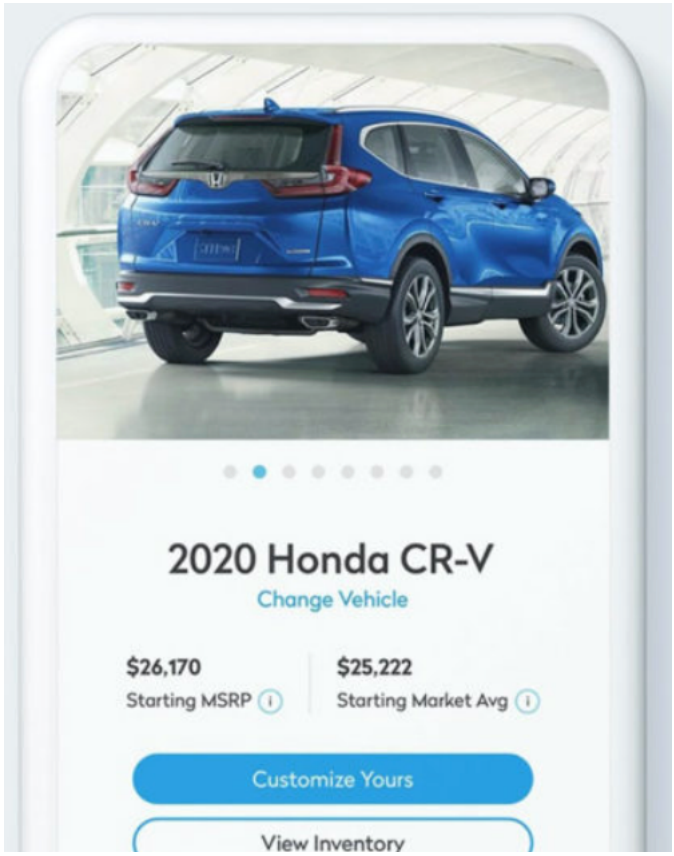

Рис. 1.9. Візуальний вигляд стартового екрану додатку TrueCar [6]

ВИСНОВОК ДО ПЕРШОГО РОЗДІЛУ

Проводячи аналіз існуючих на ринку рішень, бачимо, що додаток, який буде реалізовуватися, повинен мати такий перелік функцій:

- Відображення списку всіх транспортних засобів, наявних в базі даних
- Створення облікового запису користувача
- Вхід користувачем в свій обліковий запис
- Фільтрація авто
- Можливість розміщувати власне авто для продажу
- Можливість видаляти власне оголошення
- Фільтрація оголошень модератором

РОЗДІЛ 2
ВИБІР ЗАСОБІВ РОЗРОБКИ І АРХІТЕКТУРИ ПРОЕКТУ

2.1. Вибір засобів розробки

2.1.1. Мова Kotlin

Kotlin - це крос-платформна, статично набрана мова загального призначення з програмуванням типу. Kotlin розроблений для повноцінного взаємодії з Java, і версія JVM стандартної бібліотеки Kotlin залежить від бібліотеки класів Java, але висновок типу дозволяє синтаксису бути більш стислим. Kotlin головним чином орієнтована на JVM, але також компілює в JavaScript (наприклад, для веб-програм, що використовують інтерфейс, що використовує React) або власний код (через LLVM), для власних додатків iOS, що діляться бізнес-логікою з програмами Android. Витрати на розвиток мови несе JetBrains, тоді як Фонд Котліна захищає торгову марку Котлін.

7 травня 2019 року Google оголосив, що мова програмування Kotlin тепер є найкращою мовою для розробників додатків для Android. З моменту випуску Android Studio 3.0 у жовтні 2017 року Kotlin був включений як альтернатива стандартному компілятору Java. Компілятор Android Kotlin за замовчуванням націлений на Java 6, але дозволяє програмісту вибрати націлювання на Java 8 до 13 для оптимізації або більшої кількості функцій.

Розробка під мобільні пристрої Android стала орієнтованою на Котлін з часів Google I/O у 2019 році.

Використовуючи Kotlin для розробки під Android, ви досягнете таких цілей:

- Менше коду в поєднанні з більшою читабельністю. Витратьте менше часу на написання коду та працю, щоб зрозуміти код інших.

- Зріла мова та середовище. З часу свого створення в 2011 році Котлін постійно розвивався не лише як мова, але й як ціла екосистема із надійними інструментами. Зараз він легко інтегрований в Android Studio і активно використовується багатьма компаніями для розробки програм для Android.

- Підтримка Kotlin в Android Jetpack та інших бібліотеках. Розширення KTX додають до існуючих бібліотек Android функції мови Kotlin, такі як програми, функції розширення, лямбди та названі параметри.

- Взаємодія з Java. Ви можете використовувати Kotlin разом із мовою програмування Java у своїх програмах, не потребуючи міграції всього коду до Kotlin.

- Підтримка розробки багатоплатформенних систем. Ви можете використовувати Kotlin для розробки не тільки Android, а й iOS, серверних платформ та веб-додатків. Насолоджуйтесь перевагами спільного використання спільного коду між платформами.

- Безпечинй код. Менше коду та краща читабельність призводять до меншої кількості помилок. Компілятор Kotlin виявляє ці залишки помилок, роблячи код безпечним.

- Легке навчання. Kotlin дуже простий у вивченні, особливо для розробників Java.

- Велика громада. Котлін має велику підтримку та багато внесків від громади, яка зростає по всьому світу. За даними Google, понад 60% з 1000 найкращих додатків у Play Store використовують Kotlin. Багато стартапів та компаній Fortune 500 вже розробили додатки для Android за допомогою Kotlin.

2.1.2. Система Android

Android - це мобільна операційна система, заснована на модифікованій версії ядра Linux та іншого програмного забезпечення з відкритим кодом, призначена в основному для сенсорних мобільних пристроїв, таких як смартфони та планшети. Android розробляється консорціумом розробників, відомим як Open Handset Alliance, і комерційно фінансується Google. Він був представлений у листопаді 2007 року, а перший комерційний пристрій Android був запущений у вересні 2008 року.

Це безкоштовне програмне забезпечення з відкритим кодом; його вихідний код відомий як Android Open Source Project (AOSP), який в основному ліцензується за ліцензією Apache. Однак більшість пристроїв

Android постачаються з попередньо встановленим запатентованим програмним забезпеченням, зокрема Google Mobile Services (GMS), яке включає такі основні програми, як Google Chrome, платформа цифрового розповсюдження Google Play та пов'язана з ними платформа розробки служб Google Play. Близько 70 відсотків смартфонів Android працюють в екосистемі Google; конкуруючі екосистеми та форки Android включають Fire OS (розроблена Amazon) або LineageOS. Однак назва та логотип "Android" є товарними знаками Google, які встановлюють стандарти щодо обмеження використання "несертифікованих" пристроїв поза їх екосистемою для використання бренду Android.

Вихідний код був використаний для розробки варіантів Android для цілого ряду іншої електроніки, таких як ігрові приставки, цифрові камери, портативні медіаплеєри, ПК та інші, кожен із яких має спеціалізований користувальницький інтерфейс. Деякі відомі похідні версії включають Android TV для телевізорів та Wear OS для носимих пристроїв, розроблені Google. Пакети програм на Android, які використовують формат APK, зазвичай розповсюджуються через власні магазини додатків, такі як Google Play Store, Samsung Galaxy Store та Huawei AppGallery, або платформи з відкритим кодом, такі як Aptoide або F-Droid.

Android є найбільш продаваною ОС у світі на смартфонах з 2011 року, а на планшетах - з 2013 року. Станом на травень 2017 року вона має понад два мільярди активних користувачів щомісяця, що є найбільшою встановленою базою будь-якої операційної системи, а станом на серпень 2020 року Google У Play Store представлено понад 3 мільйони додатків.

Користувальницький інтерфейс Android за замовчуванням в основному заснований на прямих маніпуляціях, використовуючи сенсорні входи, які вільно відповідають дійсним діям, таким як гортання, натискання, стискання та зворотне стискання для маніпулювання об’єктами на екрані, разом з віртуальною клавіатурою. Ігрові контролери та повнорозмірні фізичні клавіатури підтримуються через Bluetooth або USB. Реакція на введення користувача розроблена негайно і забезпечує плавний сенсорний інтерфейс,

часто використовуючи вібраційні можливості пристрою для забезпечення тактильного зворотного зв’язку з користувачем. Внутрішнє обладнання, таке як акселерометри, гіроскопи та датчики наближення, використовуються деякими програмами для реагування на додаткові дії користувача, наприклад, регулювання екрана від портретного до альбомного залежно від того, як орієнтований пристрій, або дозволяючи користувачеві керувати транспортним засобом у гоночна гра шляхом обертання пристрою, що імітує управління кермом.

Пристрої Android завантажуються на головний екран, основний навігаційно-інформаційний «хаб» на пристроях Android, аналогічно робочому столу, що знаходиться на персональних комп’ютерах. Домашні екрани Android зазвичай складаються з піктограм додатків та віджетів; піктограми програм запускають відповідну програму, тоді як віджети відображають оновлюваний вміст, який автоматично оновлюється, наприклад прогноз погоди, поштову скриньку користувача або індикатор новин безпосередньо на головному екрані. Домашній екран може складатися з декількох сторінок, між якими користувач може проводити пальцем вперед і назад. Додатки сторонніх розробників, доступні в Google Play та інших магазинах програм, можуть широко переформатувати тему головного екрана і навіть імітувати зовнішній вигляд інших операційних систем, таких як Windows Phone. Більшість виробників налаштовують зовнішній вигляд та функції своїх пристроїв Android, щоб відрізнятись від своїх конкурентів.

У верхній частині екрана знаходиться рядок стану, що відображає інформацію про пристрій та його зв’язок. Цей рядок стану можна "потягнути" вниз, щоб відкрити екран сповіщень, де програми відображають важливу інформацію або оновлення. Сповіщення - це "коротка, своєчасна та відповідна інформація про вашу програму, коли вона не використовується", а коли натискається, користувачі переходять на екран усередині програми, що стосується сповіщення. Починаючи з Android 4.1 "Jelly Bean", "розгортаються сповіщення" дозволяють користувачеві натискати піктограму на сповіщенні,

щоб воно розгорталося та відображало більше інформації та можливі дії додатка безпосередньо з повідомлення.

На панелі сповіщень відображаються як активні, так і безшумні сповіщення.

На екрані "Усі програми" перелічено всі встановлені програми з можливістю перетягування програми зі списку на головний екран. Екран нещодавно дозволяє користувачам переключатися між нещодавно використовуваними програмами.

Додатки ("програми"), які розширюють функціональність пристроїв (і мають бути 64-розрядними), створюються за допомогою набору для розробки програмного забезпечення Android (SDK) і, часто, мови програмування Kotlin, яка замінила Java як перевагу Google мова для розробки додатків для Android в травні 2019 року, і спочатку була анонсована в травні 2017 року. Java все ще підтримується (спочатку єдина можливість для програм простору користувача і часто змішується з Kotlin), як і C ++. Java та / або інші мови JVM, такі як Kotlin, можуть поєднуватися з C / C ++, разом із вибором стандартних середовищ виконання, які забезпечують кращу підтримку C ++. Мова програмування Go також підтримується, хоча з обмеженим набором інтерфейсів прикладного програмування (API).

SDK включає повний набір засобів розробки, включаючи налагоджувач, бібліотеки програмного забезпечення, емулятор слухавки на основі QEMU, документацію, зразок коду та навчальні посібники. Спочатку підтримуваним інтегрованим середовищем розробки Google (IDE) було Eclipse з використанням плагіна Android Development Tools (ADT); У грудні 2014 року Google випустив Android Studio, засновану на IntelliJ IDEA, як основну IDE для розробки додатків для Android. Доступні інші засоби розробки, включаючи власний набір для розробки (NDK) для додатків або розширень на C або C ++, Google App Inventor, візуальне середовище для початківців програмістів та різні платформи для мобільних веб-додатків. У січні 2014 року Google представив структуру на базі Apache Cordova для перенесення веб-додатків Chrome HTML 5 на Android, загорнуті в оболонку власного додатка. Крім того,

Google придбав Firebase у 2014 році, який надає корисні інструменти для розробників програм та веб-розробників.

Android має все більший вибір сторонніх додатків, які користувачі можуть придбати, завантаживши та встановивши файл APK програми (пакет додатків Android), або завантаживши їх за допомогою програми зберігання програм, яка дозволяє користувачам встановлювати, оновлювати та видаляти програми зі своїх пристроїв. Google Play Store - це основний магазин програм, встановлений на пристроях Android, які відповідають вимогам сумісності Google та ліцензують програмне забезпечення Google Mobile Services. Google Play Store дозволяє користувачам переглядати, завантажувати та оновлювати програми, опубліковані Google та сторонніми розробниками; станом на серпень 2020 року в Play Store доступно понад три мільйони додатків для Android. Станом на липень 2013 року було встановлено 50 мільярдів додатків. Деякі оператори пропонують безпосередню оплату через оператора Google Play, де вартість програми додається до щомісячного рахунку користувача. Станом на травень 2017 року щомісяця зареєстровано понад мільярд активних користувачів Gmail, Android, Chrome, Google Play і Maps.

Через відкритий характер Android, для Android існує також низка сторонніх ринків додатків, або для того, щоб замінити пристрої, яким заборонено поставлятись із Google Play Store, або пропонувати програми, які не можуть пропонуватися в Google Play Store через до порушень політики або з інших причин. Прикладами таких сторонніх магазинів є Amazon Appstore, GetJar та SlideMe. F-Droid, інший альтернативний ринок, прагне надавати лише програми, які розповсюджуються за безкоштовними та відкритими ліцензіями.

2.1.3. Kotlin Courutines

Програми Kotlin представляють новий стиль одночасності, який можна використовувати на Android для спрощення асинхронного коду. Попри те, що вони вперше з’явились у Котліні в 1.3, концепція спільних програм існує з

самого початку програмних мов. Першою мовою, яку досліджували за допомогою програм, була Simula в 1967 році.

За останні кілька років програми стали популярнішими і тепер включені до багатьох популярних мов програмування, таких як Javascript, C#, Python, Ruby та Go, щоб назвати декілька. Програми Kotlin базуються на усталених концепціях, які використовувались для побудови великих додатків.

Отримання веб-сторінки або взаємодія з API вимагають подання мережевого запиту. Подібним чином читання з бази даних або завантаження зображення з диска передбачає читання файлу. Такі речі я називаю давно запущеними завданнями - завданнями, на які ваш додаток занадто довго зупиняється і чекає їх!

Важко зрозуміти, наскільки швидко сучасний телефон виконує код порівняно із мережевим запитом. На Pixel 2 один цикл процесора займає трохи менше 0,0000000004 секунди, що досить важко зрозуміти з людської точки зору. Однак, якщо ви розглядаєте мережевий запит як одне миготіння, приблизно 400 мілісекунд (0,4 секунди), легше зрозуміти, наскільки швидко працює центральний процесор. За один миг або трохи повільний мережевий запит центральний процесор може виконати понад один мільярд циклів!

На Android кожна програма має основний потік, який відповідає за обробку інтерфейсу користувача (наприклад, для перегляду малюнків) та координацію взаємодії користувачів. Якщо над цим потоком відбувається занадто багато роботи, програма, здається, зависає або сповільнюється, що призводить до небажаного користувацького досвіду. Будь-яке тривале завдання повинно виконуватися без блокування основного потоку, тому ваш додаток не відображатиме те, що називається "ненормальним", наприклад, заморожені анімації, або повільно реагувати на дотикові події.

Для того, щоб виконати мережевий запит від основного потоку, загальним шаблоном є зворотні виклики. Зворотні виклики забезпечують дескриптор бібліотеки, який вона може використовувати для зворотного

перегляду вашого коду в майбутньому. За допомогою зворотних викликів завантаження developer.android.com може виглядати так:

```
class ViewModel: ViewModel() {
fun fetchDocs() {
get("developer.android.com") { result ->
show(result)
}
}
}
```

Незважаючи на те, що get викликається з основного потоку, він буде використовувати інший потік для виконання мережевого запиту. Потім, як тільки результат буде доступний з мережі, зворотний дзвінок буде викликаний в основному потоці. Це чудовий спосіб вирішити тривалі завдання, і такі бібліотеки, як Retrofit, можуть допомогти вам робити мережеві запити, не блокуючи основний потік.

Спільні програми - це спосіб спростити код, який використовується для управління тривалими завданнями, такими як fetchDocs. Щоб дослідити, як спільні програми спрощують код для довготривалих завдань, перепишемо приклад зворотного виклику вище, щоб використовувати такі програми.

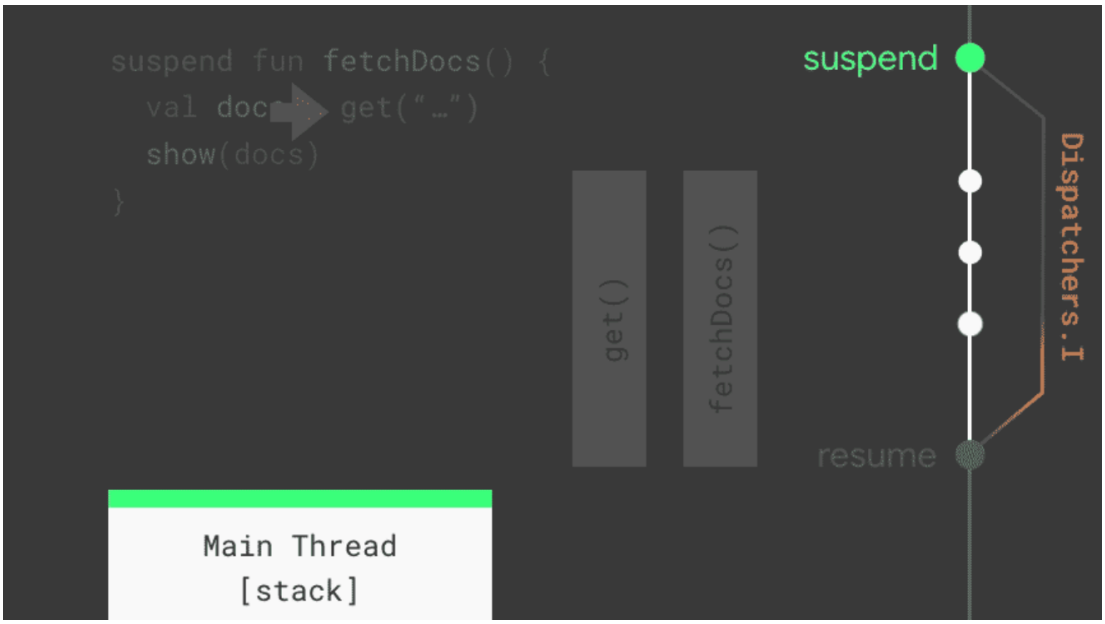

Рис. 2.1. Наглядний приклад роботи потоків з Kotlin Coroutines

У програмах Kotlin добре написані функції призупинення завжди безпечно викликати з основного потоку. Незалежно від того, що вони роблять, вони завжди повинні дозволяти будь-якому потоці викликати їх.

Але в наших додатках для Android є багато речей, які занадто повільно відбуваються в основній темі. Мережеві запити, синтаксичний аналіз JSON, читання чи запис із бази даних, або навіть просто перебір великих списків. Будь-який з них може працювати досить повільно, щоб викликати у користувача видимий «провал», і він повинен вибігти з основного потоку.

Використання suspend не говорить Kotlin про запуск функції у фоновому потоці. Варто чітко і часто говорити, що спільні програми працюватимуть по головній нитці. Насправді, це дуже гарна ідея використовувати Dispatchers.Main.immediate при запуску корутину у відповідь на подію користувацького інтерфейсу - таким чином, якщо ви в кінцевому підсумку не зробите тривале завдання, яке вимагає основної безпеки, результат може бути доступним у найближчому кадрі для користувача.

Щоб зробити функцію, яка працює надто повільно, щоб основний потік був безпечним для основного потоку, ви можете сказати програмам Kotlin виконувати роботу або за замовчуванням, або за диспетчером вводу-виводу. У Котліні всі програми повинні працювати в диспетчері, навіть коли вони працюють в основному потоці. Спільні програми можуть припинити свою діяльність, і диспетчер - це те, що знає, як їх відновити.

Щоб вказати, де повинні виконуватися програми, Kotlin надає три диспетчери, які можна використовувати для відправки потоків.

2.1.4. Бібліотека Koin

Koin - це ін'єкційна система введення залежностей, суто побудована в Котліні. Порівняно з Dagger2, ми можемо знайти багато можливостей, зручних для розробників у Koin. Крива навчання для Коїна менше порівняно з Dagger2. Є багато особливостей, які є спірними. Оскільки Койн новачок і все ще зростає, було б несправедливо порівнювати з гігантом !. Давайте розглянемо деталі реалізації Koin.

Перш ніж продовжувати, давайте пояснимо, як саме ін’єкція залежності (DI) допомагає розвитку. DI тісно пов'язаний з принципом єдиної відповідальності та інверсією залежності.

Принцип єдиної відповідальності: Клас / група класів / модуль в одній програмі відповідає за конкретну функціональність, яка вирішує одну проблему.

Інверсія залежності: Модулі верхнього рівня не повинні залежати від модулів нижчого рівня. Якщо ми дотримуватимемось цих принципів, програма матиме слабо пов'язані модулі.

Koin - це DSL, використовуючи DSL Kotlin, який описує ваші залежності в модулі та підмодулі. Ви описуєте свої визначення певними ключовими функціями, які щось означають у контексті Койна.

Я поясню ключові функції пізніше в цій публікації, але спочатку ви повинні знати, що після того, як ви описали свої залежності в модулях Koin, вам потрібно викликати функцію startKoin, яка передає Android Context, і ваш список модулів.

Визначення модуля дуже схоже на визначення Dagger-2 @Module, він служить контейнером для набору послуг, які слід надавати під час виконання. Ви поміщаєте всі послуги, які повинен надавати модуль, у блок модулів.

Фабрика служить для того, щоб визначити залежність, вона повідомляє фреймворку Koin не зберігати цей екземпляр, а навпаки, створювати новий екземпляр цього сервісу в будь-якому місці, де потрібен сервіс.

Одне визначення робить прямо протилежне тому, що робить фабрика. Це вказує Коїну зберегти екземпляр і надати цей єдиний екземпляр, де це потрібно. Це можна зробити певними способами, подібними до дефолтних, визначених вище. Це схоже на анотацію Dagger-2 @Singleton.

Функція get - це загальна функція, яка використовується для вирішення залежностей там, де це необхідно. Ви використовуєте його для вирішення будь-якої конкретної залежності, яка вам потрібна, і її можна використовувати наступним чином.

Контейнер залежностей може містити властивості конфігурації або значення, які можна прочитати та встановити під час виконання. Це робиться за допомогою функцій getProperty або setProperty. Ці властивості містяться в

контейнері пари ключ-значення, ви читаєте та пишете за допомогою ключа та вказуєте значення, яке потрібно встановити, або тип для читання.

2.1.5. Бібліотека OkHttp

HTTP - це спосіб сучасних мереж додатків. Це спосіб обміну даними та медіа. Завдяки ефективному HTTP ваш матеріал швидше завантажується та економить пропускну здатність.

OkHttp - це клієнт HTTP, який за замовчуванням ефективний:

Підтримка HTTP / 2 дозволяє всім запитам до одного хоста спільно використовувати сокет.

Пул підключення зменшує час очікування запиту (якщо HTTP / 2 недоступний).

Прозорий GZIP зменшує розміри завантажуваних файлів.Кешування відповідей дозволяє повністю уникнути мережі для повторних запитів.OkHttp наполегливо працює, коли мережа неприємна: вона мовчки оговтається від загальних проблем із підключенням. Якщо у вашої служби кілька IP-адрес, OkHttp спробує альтернативні адреси, якщо перше підключення не вдасться. Це необхідно для IPv4 + IPv6 та служб, розміщених у резервних центрах обробки даних. OkHttp підтримує сучасні функції TLS (TLS 1.3, ALPN, закріплення сертифікатів). Його можна налаштувати на широке з'єднання.

Використовувати OkHttp просто. Його API запиту / відповіді розроблений з вільними розробниками та незмінністю. Він підтримує як синхронне блокування викликів, так і асинхронні виклики із зворотними викликами.

2.2. Вибір архітектури проекту

Проект з декількома модулями Gradle відомий як багатомодульний проект. У мультимодульному проекті, який постачається у вигляді єдиного файлу .apk без функціональних модулів, загальноприйнятим є модуль програми, який може залежати від більшості модулів вашого проекту, та базовий або основний модуль, від якого зазвичай залежать інші модулі.

Модуль програми, як правило, містить ваш клас Application, тоді як базовий модуль містить усі загальні класи, спільні для всіх модулів у вашому проекті.

Модуль програми - це гарне місце для декларування компонента програми (наприклад, ApplicationComponent на зображенні нижче), який може надати об’єкти, які можуть знадобитися іншим компонентам, а також одиночні елементи програми. Наприклад, класи, такі як OkHttpClient, синтаксичні аналізатори JSON, засоби доступу до вашої бази даних або об'єкти SharedPreferences, які можуть бути визначені в основному модулі, будуть надані ApplicationComponent, визначеним в модулі програми.

2.2.1. Application Module

Ми почнемо з цього модуля, оскільки це модуль, без якого проект не зможе бути запущеним. Це, по суті, основний модуль програми, і він часто знаходиться під модулем програми, використовуючи плагін програми у файлі build.gradle для позначення цього:

```
apply plugin: 'com.android.application'
```

Тепер, якщо ви створюєте єдину модульну програму, це буде єдиний модуль типу модуля, який ви будете використовувати протягом усього проекту. У проектах з одним модулем цей базовий модуль буде містити всі обов'язки ваших програм - бути в Інтерфейсі користувача, Мережі, Кеш-пам’яті, Операції з даними - Ви вкажете це ім’я, і він буде там.

2.2.2. Core Module

Ви можете по суті використовувати ці модулі, щоб відокремити суміжні області вашого проекту, щоб вони були відокремлені від основного модуля вашого проекту. Для них насправді не є придуманим терміном, але давайте називати їх основними модулями.

Наприклад, скажімо, у вас є проект, який підтримує як знос Android, так і телефони Android, у вас може бути модуль для шару інтерфейсу Wear, шару інтерфейсу телефону, а потім основного модуля. Цей основний модуль буде містити всю спільну бізнес-логіку, операції з даними, кешування тощо, що використовується обома модулями інтерфейсу користувача. У цьому випадку

ви сприяєте повторному використанню коду, оскільки всі ці аспекти можуть бути використані між двома модулями інтерфейсу користувача.

Повторне використання - це не єдина причина того, чому ви можете створити такий модуль. Відокремлення базової логіки також може допомогти створити чітке розділення проблем, що полегшує розуміння, тестування та підтримку коду у вашій команді - у деяких випадках ви також можете скористатися вдосконаленням часу побудови.

Так, наприклад, у вас може бути велика програма зі складним рівнем даних, яка взаємодіє як з віддаленим джерелом даних, так і з кешем. У таких випадках поділ цих частин логіки на власні модулі, що відповідають конкретній відповідальності, дозволяє досягти цього. Це може призвести до того, що ми будемо мати модуль даних (для обробки даних та джерела даних), віддалений модуль (для обміну даними з джерелом даних) та кеш-модуль (для обробки локальної стійкості даних) поряд із модулем презентації для обробки користувачем- облицювальні частини нашого додатку.

Немає чітких вказівок щодо того, що повинно бути в цих основних модулях. Моя порада полягала б у тому, щоб не переносити речі в модуль заради цього, але якщо ви відчуваєте, що для вашої команди буде певна користь (наприклад, згадані вище), то перенесення відповідальності в модуль може допомогти вам досягти деяких переваги модуляризації.

Основні модулі, що містять посилання на фреймворк Android, повинні використовувати плагін бібліотеки:

```
apply plugin: 'com.android.library'
```

2.2.3. Abstraction Modules

Основні модулі часто можуть використовуватися для переміщення фрагментів спільної логіки або незалежних обов’язків, але іноді ми можемо захотіти абстрагувати певну відповідальність третьої сторони з нашого додатку. Це часто може бути гарною ідеєю, щоб наш додаток не був щільно пов'язаний з конкретною реалізацією чогось - у цій ситуації ми просто спілкувалися б через інтерфейс між програмою та модулем абстракції, дозволяючи нам легко вимкнути реалізацію для інший, якщо потрібно. Тому,

якщо якась бібліотека застаріла, послуга вимкнеться або з якоїсь причини вам потрібно змінити реалізацію, ви можете зробити це з мінімальними перервами в основі коду.

У цих випадках ваші модулі абстракції будуть використовувати плагін бібліотеки android або просто будуть чистим модулем kotlin / java (залежно від залежностей бібліотеки), який ми розглядали раніше.

2.2.4. Abstraction Modules

Основні модулі часто можуть використовуватися для переміщення фрагментів спільної логіки або незалежних обов’язків, але іноді ми можемо захотіти абстрагувати певну відповідальність третьої сторони з нашого додатку. Це часто може бути гарною ідеєю, щоб наш додаток не був щільно пов'язаний з конкретною реалізацією чогось - у цій ситуації ми просто спілкувалися б через інтерфейс між програмою та модулем абстракції, дозволяючи нам легко вимкнути реалізацію для інший, якщо потрібно. Тому, якщо якась бібліотека застаріла, послуга вимкнеться або з якоїсь причини вам потрібно змінити реалізацію, ви можете зробити це з мінімальними перервами в основі коду.

У цих випадках ваші модулі абстракції будуть використовувати плагін бібліотеки android або просто будуть чистим модулем kotlin / java (залежно від залежностей бібліотеки), який ми розглядали раніше.

ВИСНОВОК ДО ДРУГОГО РОЗДІЛУ

Після досліження найкращих практик розробки програмного коду під платформу Android, я обрав багатомодульну архітектуру додатку, де кожен окремий модуль буде відповідати за окремий набір функціоналу. Всередині себе, кожен модуль буде побудований, опираючись на шаблон MVVM.

Для зручності обробки багатопотоковості, обрана технологія Kotlin Coroutines.

Для обробки REST запитів до серверу була обрана бібліотека OkHttp.

Впровадження залежностей буде реалізовано, використовуючи бібліотеку Koin.

Загалом, програмний код буде написаний на мові програмування Kotlin.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ДОДАТКУ

Розглянемо архітектуру всередині модулів додатку. За основу обраний паттерн розробки MVVM.

1.1. Model

Модель - це невізуальний клас, який має дані для використання. Приклади включають DTO (об'єкти передачі даних), POJO (звичайні об'єкти Java) та об'єкти сутності. Це загальновживаний сервіс або сховище, які потребують доступу або кешування даних.

3.1.1 Джерело інформації для обробки даних про неавторизованого покупця системи.

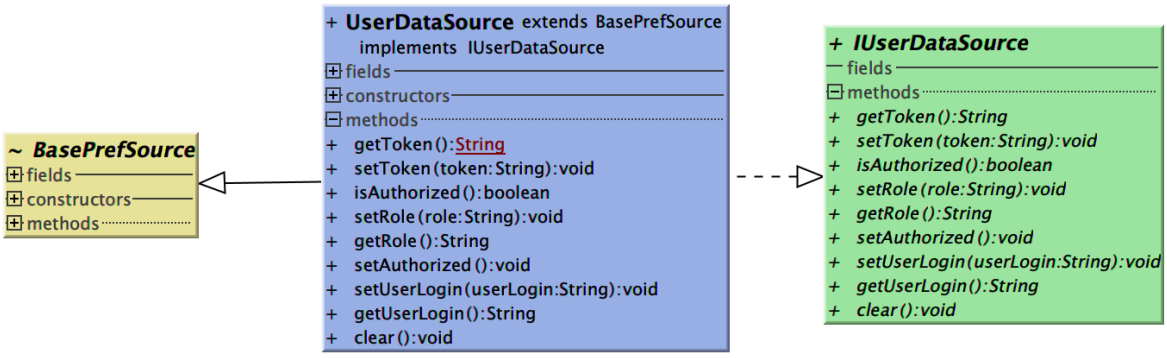

Рис. 3.1. Джерело для отримання інформації про незалогіненого покупця системи

Є інтерфейс UserDataSource, який наслідує клас BasePrefSource для зручної роботи з локально зберігаємими даними про користувача та сесію роботи з сервером, а також цей клас релізовує інтерфейс IUserDataSource, задля зручної роботи з ним, як з джерелом обміну даними додатку з сервером. Розглянемо, який функціонал міститься в даному класі:

Таблиця 3.1

Опис функції класу UserDataSource:

Назва функції	**Опис функції**
String getToken()	Отримати токен сесії, необхідний для отримання даних про автомобілі.
void setToken(String token)	Встановити токен сесії користувача.
boolean isAuthorized()	Отримати стан логіну користувача (true, якщо користувач залогінений і false, якщо не залогінений).
void setAuthorized (boolean isAuthorized)	Встановити стану логіну користувача
String getRole()	Отримати роль користувача. Можливі варіанти – ADMIN і USER. Зажежно від ролі користувача, він матиме різний набір доступних йому функцій додатку.
void setRole(String role)	Встановити роль залогіненого користувача.
String getUserLogin()	Отримати логін користувача, необхідний для спілкування між користувачами.
void setUserLogin(String userLogin)	Встановити логін залогіненого користувача.
void clear()	Очистити всі локальні дані про користувача в разі лог-ауту.

3.1.2. Джерело інформації для обробки даних про авторизованого покупця системи.

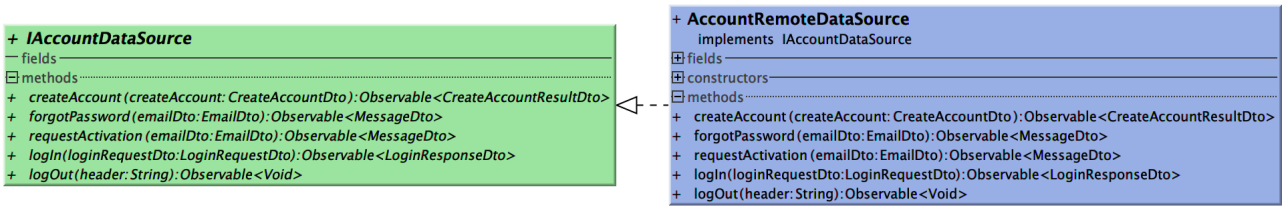

Рис. 3.2. Джерело даних для отримання даних про залогіненого користувача

Існує інтерфейс IAccountDataSource, який використовується для роботи з незалогіненим користувачем. Оскільки весь процес такої роботи повинен проходити лише за наявності доступного обміну даними між клієнтом і сервером, інтерфейс має лише однин клас, який його реалізовує – AccountRemoteDataSource.

Таблиця 3.2

Опис функції класу AccountRemoteDataSource:

Назва функції	**Опис функції**
Observable<CreateAccountResultDto> createAccount(CreateAccountRequestDto create account request)	Створити нового користувача в системі.
Observable<MessageDto> forgotPassword(EmailDto email)	Відновити пароль у разі, якщо користувач його втратив.
Observable<LoginResultDto> logIn(LoginRequestDto loginRequestDto)	Вхід користувача в систему
Observable<LogoutResultDto> logOut(LogoutRequestDto loginRequestDto)	Вихід користувача з системи

3.1.3. Джерело інформації для обробки даних про транспортні засоби.

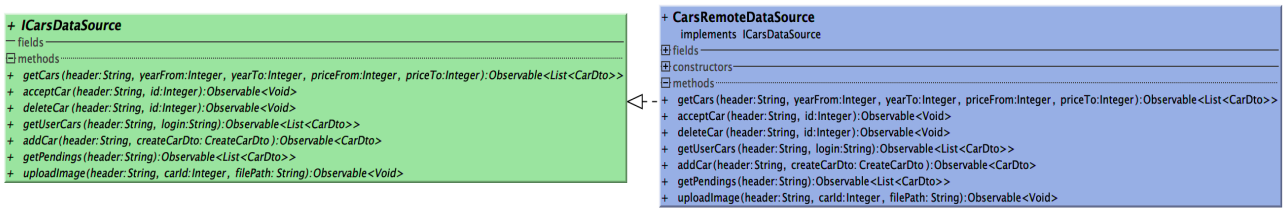

Рис. 3.3. Джерело даних для отримання даних про автомобілі

Існує інтерфейс ICarsDataSource, який використовується для роботи з автомобілями. Оскільки весь процес такої роботи повинен проходити лише за наявності доступного обміну даними між клієнтом і сервером, інтерфейс має лише однин клас, який його реалізовує – CarsRemoteDataSource. Розглянемо, який функціонал міститься в даному класі:

Таблиця 3.3. Опис функції класу CarsRemoteDataSource:

Назва функції	**Опис функції**
Observable<List<CarDto>> getCars (String header, int year from, int yearTo, int priceFrom, int priceTo)	Отримати список доступних автомобілів за вибраними параметрами. Якщо параметри не будуть передані серверу, отримати всі доступні авто
Observable<Void> acceptCar(String header, int id)	Підтвердження авто модератором для того, щоб воно попало в список всіх авто.
Observable<Void> removeCar(String header, int id)	Видалення авто модератором для того, щоб воно не попало в список доступних автомобілів звичайним користувачам. Доступно тільки модератору.
Observable<List<CarDto>> getUserCars (String header, String login)	Отримати оголошення про продаж автомобілів, створені користувачем

Продовження таблиці 3.3.

Назва функції	**Опис функції**
Observable<CarDto> addCar(String header, CreateCarDto createCarDto)	Додати авто для перевірки його модератором сервісу. Доступно всім користувачам.
Observable<List<CarDto>> getPendings (String header)	Отримати список оголошень, які необхідно переглянути модераторові для підтвердження. Доступно тільки модератору.
Observable<Void> uploadImage(String header, int id, String filePath)	Завантажити фотографію автомобіля.

Розглянемо моделі даних, присутні в додатку

Таблиця 3.4

Моделі даних програмного додатку

Назва моделі	**Призначення моделі**
BaseEntity	Базова модель
AccountRole	Роль користувача. ADMIN – модератор, USER – звичайний користувач
MessageDto	Модель обміну провідомленнями для виводу повідомлень з серверу на екран
ErrorDto	Модель помилки, яка використовуєьться при виникненні будь-якої помилки на сервері при роботі з ним

Продовження таблиці 3.4

Назва моделі	**Призначення моделі**
CreateCarDto	Модель, яка використовується для додання оголошення про продаж авто.
CarDto	Модель автомобіля.
LoginResponseDto	Модель, яка отримується при успішному вході користувача в систему.
LoginRequestDto	Модель, яка відправляється на сервер для спроби входу користувача в систему.
EmailDto	Модель передачі email користувача на сервер
CreateAccountResultDto	Модель, яка використовується як результат створення облікового запису користувача
CreateAccountDto	Модель, яка використовується для створення облікового запису користувача
AccountDto	Модель облікового запису користувача

1.2. ViewModel

В частину ViewModel входять класи, які відповідають за бізнес логіку додатку. В зробленому додатку існує базовий клас ViewModel, який містить в собі функції прив’язки activity до нього, а також для обробки підписок на взаємодію з бекендом, які використовуються в конкретному viewmodel. Розглянемо ці функції деталь:

1.2.1. BaseViewModel

Таблиця 3.5

Опис функцій класу BaseViewModel:

Назва функції	**Опис функції**
void addSubscription(Subscription subscription)	Додати підписку на роботу з сервером
void clear()	Очистити всі дані класу (об'єкт View та підписки для роботи з сервером)
Void setView(View view)	Встановити об'єкт View для поточного презентера

Для всіх окремих слоїв бізнес-логіки програми додано власний клас-viewmodel, який реалізує клас BaseViewModel. Ці класи описані нище.

1.2.2. AddCarViewModel

Цей клас додано для роботи з бекендом для створення оголошення для продаж свого транспортного засобу.

Таблиця 3.6

Опис функцій класу AddCarViewModel:

Назва функції	**Опис функції**
void addCar(String header, CreateCarDto createCarDto)	Додати оголошення про продаж власного авто
void uploadPhoto (String header, int carId, String filePath)	Завантажити фото авто, яке продається

1.2.3. CarListViewModel

Цей клас створено для отримання списку доступних автомобілів. Використовується як для звичайних користувачів, так і для модераторів.

Таблиця 3.7

Опис функцій класу CarListViewModel:

Назва функції	**Опис функції**
void getCars(String header, int year from, int yearTo, int priceFrom, int priceTo)	Отримати список автомобілів за заданими параметрами. Якщо параметри не будуть передані, повернеться список з усіма доступними користувачеві автомобілями
void getMyCars(String header, String userLogin)	Отримати список оголошень, створених поточним користувачем
void getPendings(String header)	Отримати список оголошень, які потребують перевірки модератором. Доступно лише модераторові.

1.2.4. ForgotPasswordViewModel

Цей клас додано для обробки бізнес-логіки, яка відповідає за відновлення паролю користувача.

Таблиця 3.8

Опис функцій класу ForgotPasswordViewModel:

Назва функції	**Опис функції**
void onForgotPassword(EmailDto emailDto)	Отримати лист на пошту для відновлення паролю.

1.2.5. FullCarInfoViewModel

Цей клас створено для обробки бізнес-логіки, необхідної для підтвердження оголошення модератором перед тим, як воно потрапить у список актуальних оголошень, або видалення оголошення модератором або користувачем.

Таблиця 3.9

Опис функцій класу FullCarInfoViewModel:

Назва функції	**Опис функції**
void acceptCar(String header, int carId)	Підтвердити оголошення для того, щоб воно потрапило у актуальні оголошення і стало доступним всім користувачам. Цей функціонал доступний лише модератору.
void removeCar(String header, int carId)	Видалити оголошення. Цей метод доступний модератору для всіх оголошень і звичайному користувачеві для оголошень, створених ним самим.

1.2.6. LogInViewModel

Цей клас створено для обробки бізнес-логіки, необхідної для входу користувача в обліковий запис, а також для повторного запиту на активацію облікового запису, якщо він не був активований впродовж 24 годин після реєстрації.

Таблиця 3.10

Опис функцій класу LoginViewModel:

Назва функції	**Опис функції**
void resendValidation(EmailDto emailDto)	Надіслати лист для повторної активації облікового запису.
void login(LoginRequestDto loginRequestDto)	Увійти в систему

1.2.7. SignUpViewModel

Цей клас створено для обробки бізнес-логіки, необхідної для створення облікового запису користувача.

Таблиця 3.11

Опис функцій класу SignUpPresenter:

Назва функції	**Опис функції**
void signUp(CreateAccountDto createAccountDto)	Створити обліковий запис користувача.

1.3. View

View – це модуль, який відповідає за відображення графічного інтерфейсу користувача в залежності від подій (успіх, помилка і т.д.), які обробляються в ViewModel'і.

1.3.1. BaseView

В програмного додатку до цієї роботи доданий базовий клас View, який наслідується кожним класом, який відповідає за відображення графічного інтерфейсу користувача. Розглянемо, який функціонал закладений в базовий клас BaseView:

Таблиця 3.11

Опис функцій інтерфейсу BaseView:

Назва функції	**Опис функції**
void showError(String error)	Відобразити помилку
Context getContext()	Отримати контекст додатку для отримання його ресурсів у місці, це використовується поточний об'єкт View

1.3.2. AddVehicleView

Цей клас був створений для відображення інтерфейсу, який відображається користувачу коли доданні оголошення про продаж автомобіля.

Таблиця 3.12

Опис функцій інтерфейсу AddCarView:

Назва функції	**Опис функції**
void onAdded(CarDto carDto)	Відобразити графічний інтерфейс при успішному додаванні оголошення в базу даних сервера.

1.3.3. VehicleCollectionView

Цей клас був створений для відображення інтерфейсу, який відображається користувачу при перегляді всіх доступних йому оголошень про продаж автомобілів.

Таблиця 3.13

Опис функцій інтерфейсу CarListView:

Назва функції	**Опис функції**
void showCarList(List<CarDto> cars)	Відобразити графічний інтерфейс доступних користувачеві оголошень про продаж автомобілів

1.3.4. ForgotPasswordView

Цей клас був створений для відображення інтерфейсу, який відображається користувачу у випадку відновлення паролю.

Таблиця 3.14

Опис функцій інтерфейсу ForgotPasswordView:

Назва функції	**Опис функції**
void showSuccessDialog(MessageDto messageDto)	Показати діалог про успішне відправлення листа для відновлення паролю на електронну пошту користувача.

1.3.5. FullVehicleInfoView

Цей клас був створений для відображення інтерфейсу, який відображається користувачу на екрані з повною інформацією про автомобіль.

Таблиця 3.15

Опис функцій інтерфейсу FullCarInfoView:

Назва функції	**Опис функції**
void onRemoved()	Обробити подію про видалення авто у графічному інтерфейсі користувача.
void onAccepted()	Обробити подію про підтвердження авто модератором у графічному інтерфейсі користувача.

1.3.6. SignInView

Цей клас був створений для відображення інтерфейсу, який відображається користувачу на екрані для входу в обліковий запис користувача.

Таблиця 3.16

Опис функцій інтерфейсу LogInView:

Назва функції	**Опис функції**
void processLogginIn(LoginResponseDto loginresponse)	Показати користувачу екран успішного входу в систему.

1.3.7. CreateAccountView

Цей клас був створений для відображення інтерфейсу, який відображається користувачу на екрані для створення облікового запису користувача.

Таблиця 3.17

Опис функцій інтерфейсу SignUpView:

Назва функції	**Опис функції**
void startApplication(CreateAccountResultDto createAccountResultDto)	Показати користувачеві екран успішної реєстрації в системі.

1.4. Взаємодія додатку з сервером

Робота додатку з сервером через мережу інтернет реалізована через рекомендовану до використання android-розробниками компанії Google бібліотеку Retrofit 2.0. В джерела даних об'єкт класу Retrofit потрапляє через реалізований в проекті DI (dependency injection) за допомогою бібліотеки Dagger 2. Реалізація асинхронних викликів реалізована через фреймворк RxJava. Детальний опис кожного з вищеперерахованих компонентів можна прочитати в другому розділі цієї роботи.

Розглянемо, які методи серверу використовує програмний додаток.

1.4.1. VehicleApiSet

Далі будуть описані методи для взаємодії фронтенду і бекенду.

Таблиця 3.18

Моделі функцій інтерфейсу SignUpView:

Тип REST-запиту	**Відносний шлях до методу серверу**	**Опис методу сервера**
GET	api/cars	Отримання доступних користувачеві оголошень
GET	api/activate	Підтвердження оголошення модератором
GET	api/delete	Видалення оголошення
GET	api/cars/user	Отримання оголошень конкретного користувача
POST	api/add	Додавання нового оголошення
GET	api/disabledCars	Отримання оголошень, які потребують перевірки модератором
POST	api/car/{carId}/image	Завантаження фотографії автомобіля.

1.4.2. UserHandlingApiSet

Нижче описані методи для стоворення облікового запису і взаємодії з ним.

Таблиця 3.19

Моделі функцій інтерфейсу UserHandlingApiSet:

Тип REST-запиту	**Відносний шлях до методу серверу**	**Опис методу сервера**
POST	user	Створення облікового запису користувача.
POST	user/forgotPassword	Відновлення паролю.
POST	user/requestActivation	Створення запиту на можливість повторної активації облікового запису користувача.

ВИСНОВОК ДО ТРЕТЬОГО РОЗДІЛУ

При реалізації програмного додатку роботи була застосована архітектура MVVM у поєднанні з впровадженням залежностей (dependency injection).

Було розроблено три основних модуля – Model, View, ViewModel, опис яких знаходиться вище.

В програмний додаток додано багато бібліотек, які полегшують життя розробника. Серед таких – Glide, Android App Compat бібліотеки, ButterKnife і т.д.

Як і задумувалось, для викликів web-методів серверу, була застосована бібліотека Retrofit 2. Для опису асинхронних викликів був використаний фреймворк для реактивного програмування Kotlin Couroutines. Для впровадженням залежностей застосована бібліотека Dagger 2.

РОЗДІЛ 4
ІНСТРУКЦІЯ КОРИСТУВАЧА І ТЕСТУВАННЯ ДОДАТКУ

Розглянемо кожен екран додатку у порядку, в якому його бачитиме користувач.

4.1. Екран запуску додатку

На старті програми користувачеві системи показується початковий екран з лого продукту.

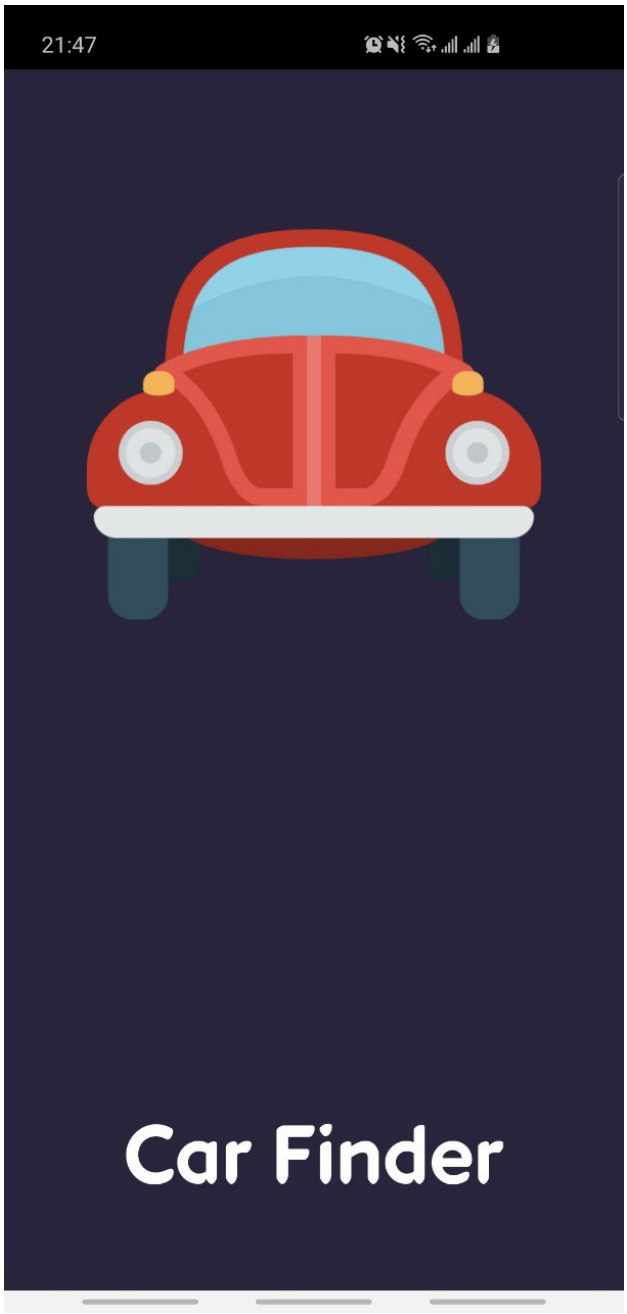

Рис. 4.1. Стартовий екран додатку, відображення логотипу сервісу

4.2. Екран з описом ключових можливостей та особливостей системи

Далі користувач попадає на екран з коротким описом можливостей додатку, які перелистуються горизонтально, під ними розміщений індикатор, який показує, яка саме підказка про можливості додатку відкрита на екрані в поточний момент часу.

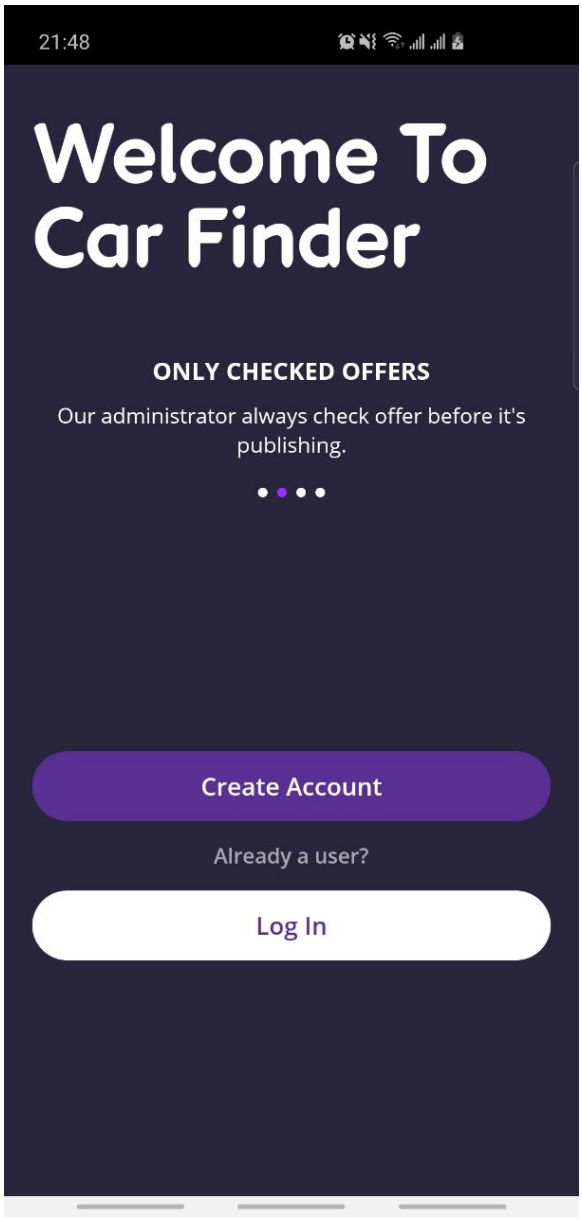

Рис. 4.2. Вітання додатку, короткий опис можливостей сервісу

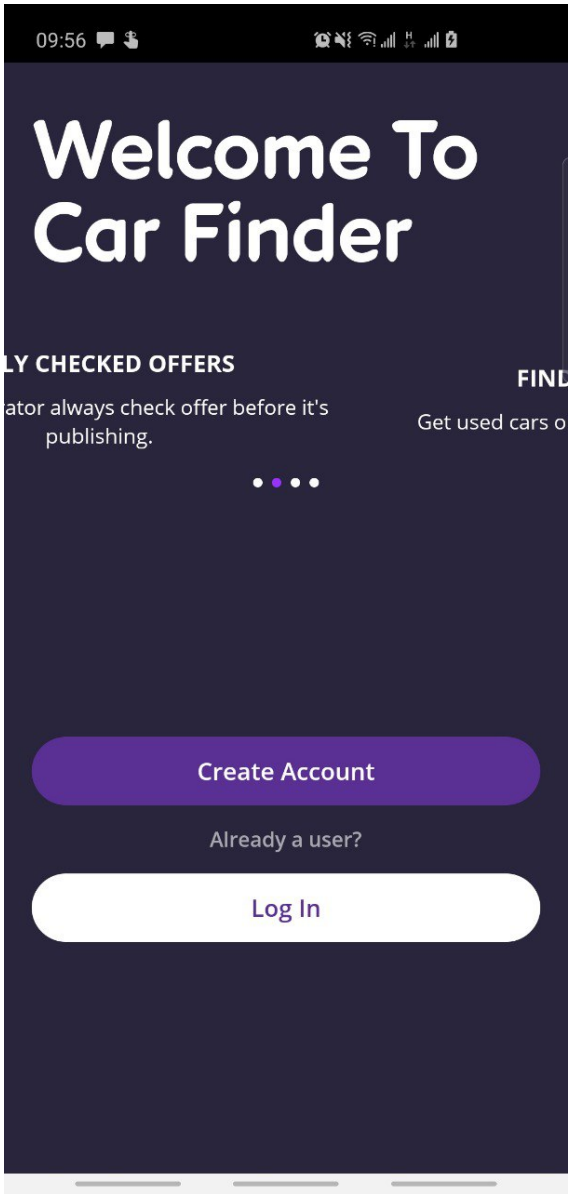

Рис. 4.3. Вітання додатку, короткий опис особливостей продукту, зміна короткого опису можливостей

Перелистування опису коротких можливостей додатку відбувається в горизонтальній осі скролу.

Також на поточному екрані розміщені кнопки для входу в систему або для створення нового облікового запису користувача.

Після цього екрану користувач може попасти на екран створення нового облікового запису або на екран входу в систему, залежно від його вибору на поточному екрані.

4.3. Екран створення акаунту користувача

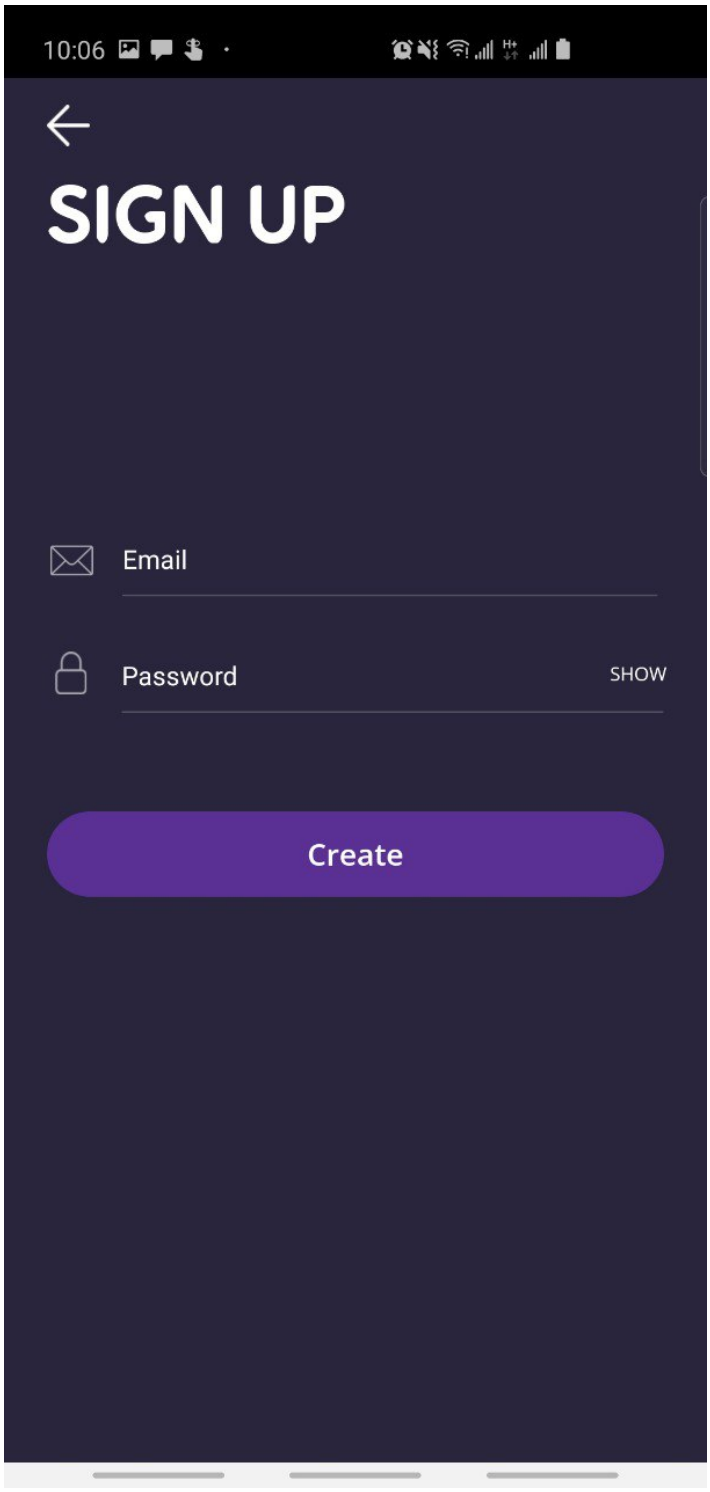

Рис. 4.4. Створення нового облікового запису користувача

Для реєстрації користувачеві необхідно і достатньо заповнити поля email і паролю. Пароль повинен бути як мінімум 8 символів довжиною, вміщувати хоча б одну букву великого алфавіту і хоча б одну букву малого алфавіту, хоча

б одну цифру і хоча б один спеціальний символ. Ці підказки відкриваються користувачеві, коли він починає вводити пароль в поле вводу.

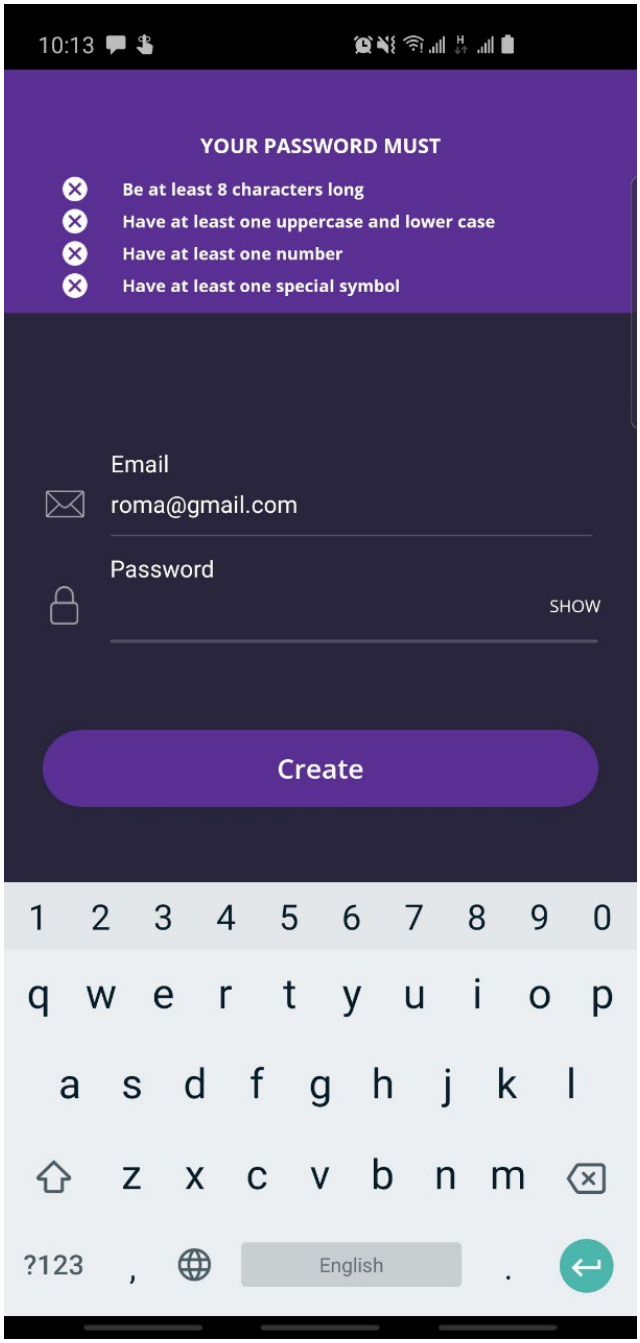

Рис. 4.5. Створення нового облікового запису користувача. Валідація паролю

Коли введений користувачем пароль відповідає якомусь з необхідних критеріїв, це відображається на екрані у вигляді пройденого «чекпоїнту» в верхній частині екрану.

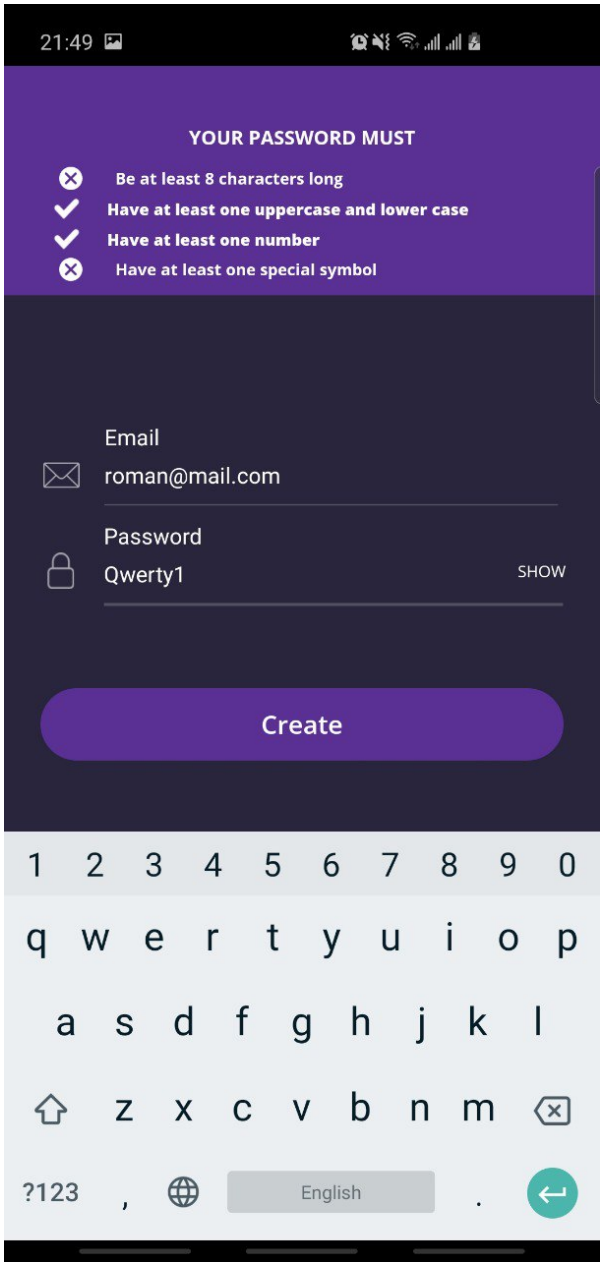

Рис. 4.6. Створення нового облікового запису користувача. Валідація паролю

Пароль можна заховати за зірочками або показати користувачеві шляхом натиснення кнопки “SHOW” біля поля введення паролю. У разі успішного створення облікового запису користувачеві виводиться наступне діалогове вікно:

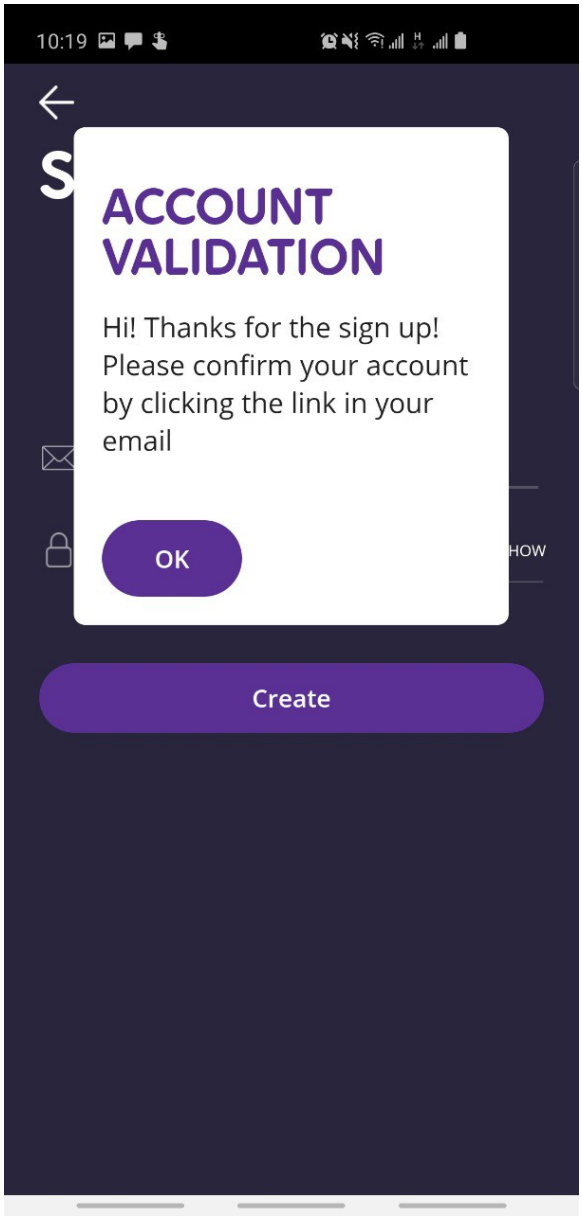

Рис. 4.6. Створення нового облікового запису користувача. Успіх

Для підтвердження облікового запису користувачеві необхідно відкрити пошту, вказану при реєстрації, і перейти за посиланням, доданим в листі. Після натискання користувачем на кнопку “OK” в діалоговому вікні, він потрапить на екран входу в систему.

4.4. Екран авторизації в платформу

Для авторизації в платформу покупцеві потрібні вказати email і пароль, вказані при створенню облікового запису.

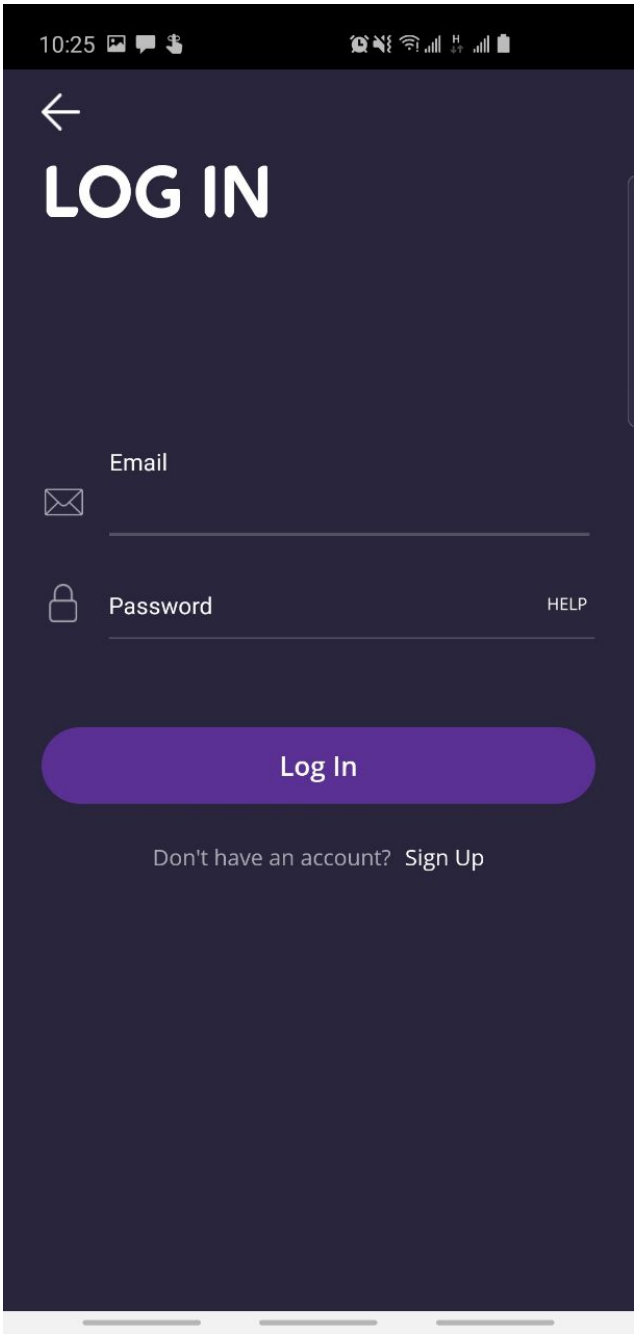

Рис. 4.7. Вхід користувачем в платформу

З цього екрану користувач може, за необхідності, перейти на екран створення облікового запису, натиснувши на кнопку “Sign Up”. У разі, якщо користувач втратив свій пароль, він може натиснути на кнопку “HELP” біля поля для вводу паролю, тоді він потрапить на екран відновлення паролю.

4.5. Екран відновлення паролю облікового запису

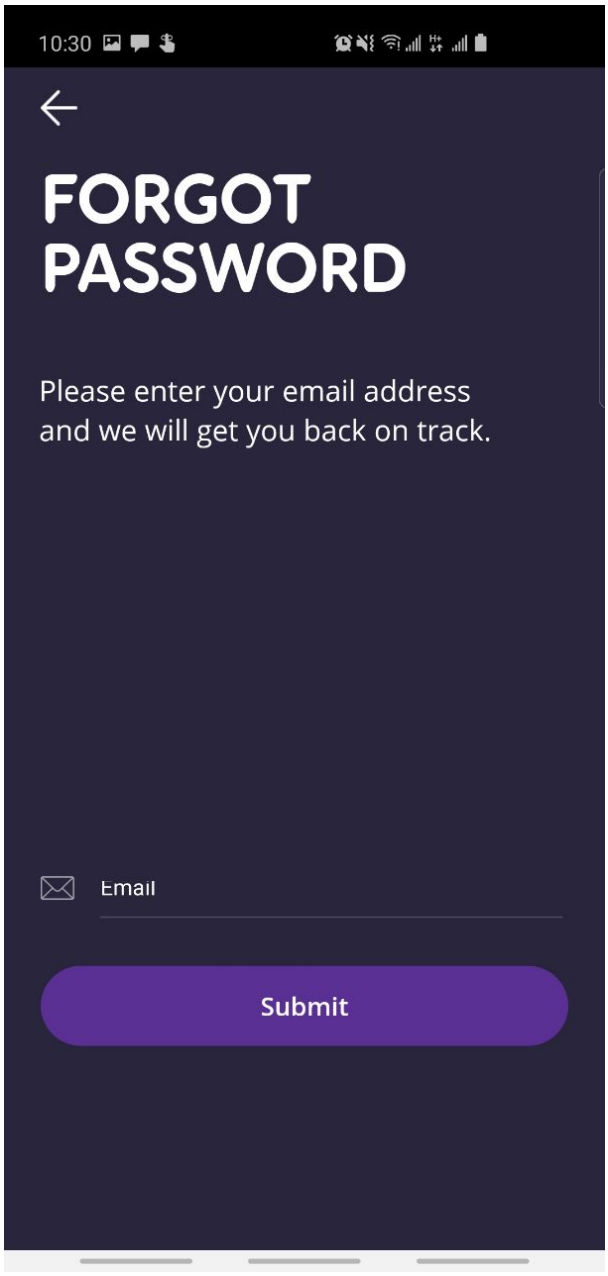

Рис. 4.8. Відновлення паролю

Тут користувачеві необхідно ввести email, вказаний при рєстрації, тоді йому на пошту буде відправлений лист з посиланням, перейшовши на яке він зможе встановити новий пароль.

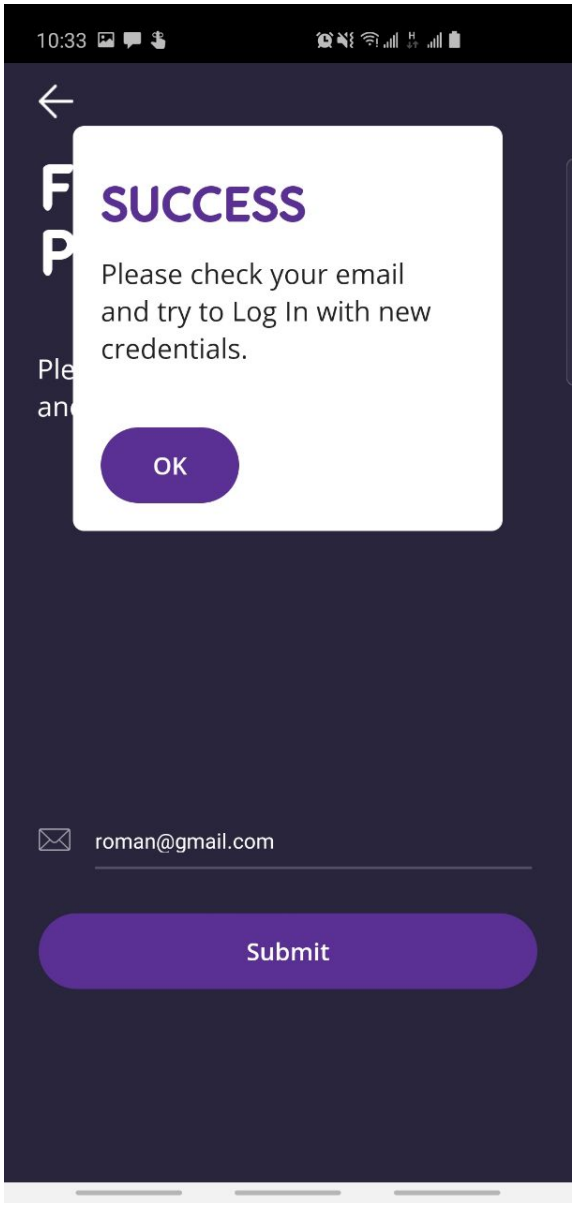

Рис. 4.9. Відновлення паролю. Успішний випадок

4.6. Екран пошуку оголошень про продаж транспортного засобу

Після авторизації в платформу покупець попадає на екран пошуку оголошень.

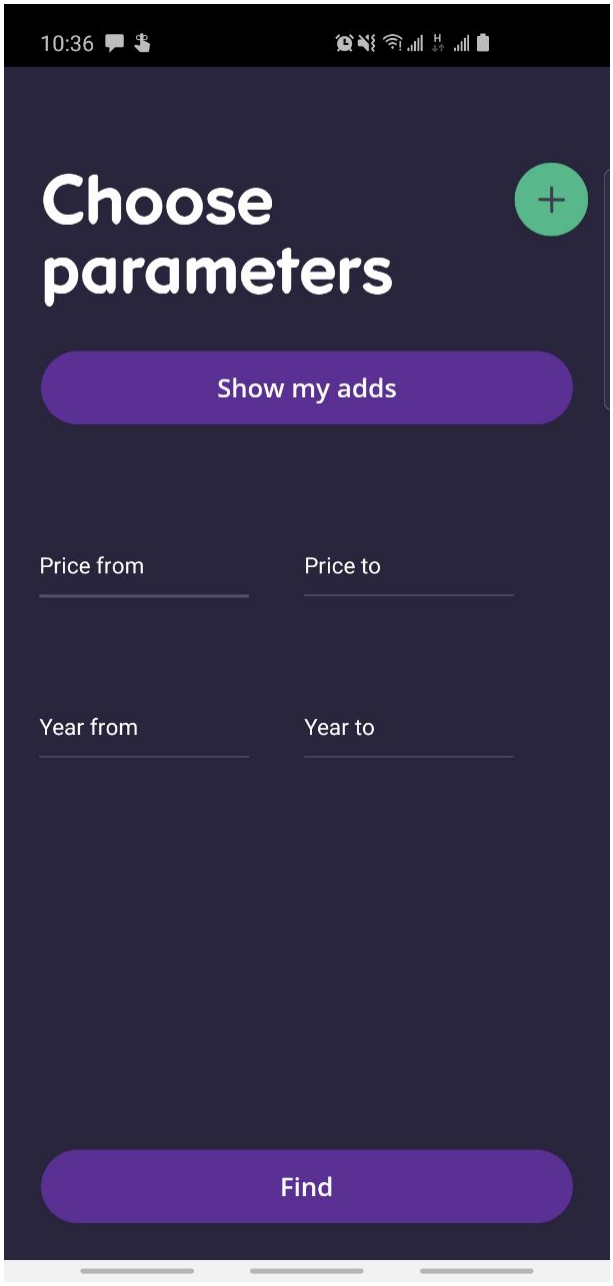

Рис. 4.10. Пошук оголошень. Екран звичайного користувача

Такий вигляд має екран пошуку оголошень звичайного користувача. Якщо у систему ввійшов модератор, на цьому екрані у нього ще буде кнопка для відображення оголошень, які потребують перевірки.

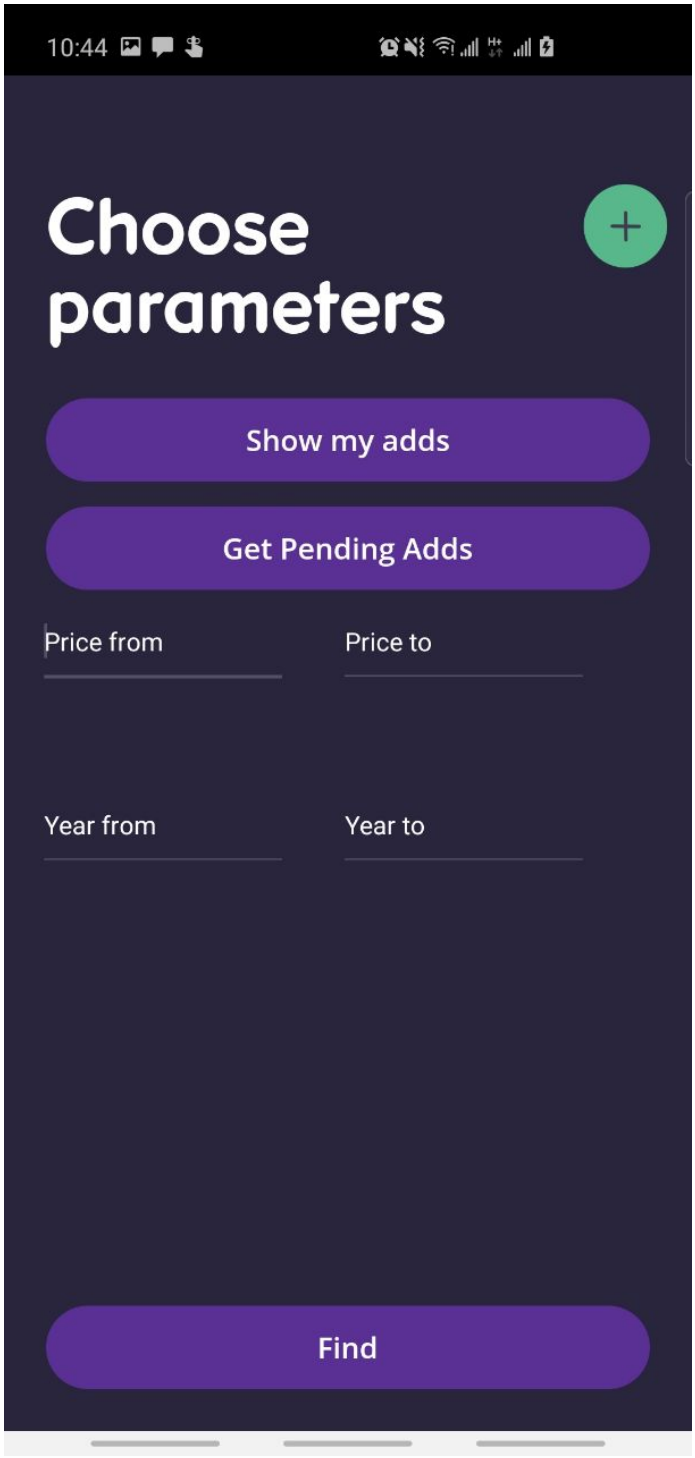

Рис. 4.11. Пошук оголошень. Екран модератора

Ця кнопка потрібна для того, щоб модератор міг завантажити оголошення, які він повинен перевірити перед тим, як вони потраплять у список всіх доступних оголошень.

На даному екрані користувач може вказати параметри пошуку оголошень, які його цікавлять. Доступні параметри – цінові межі та межі року випуску автомобіля.

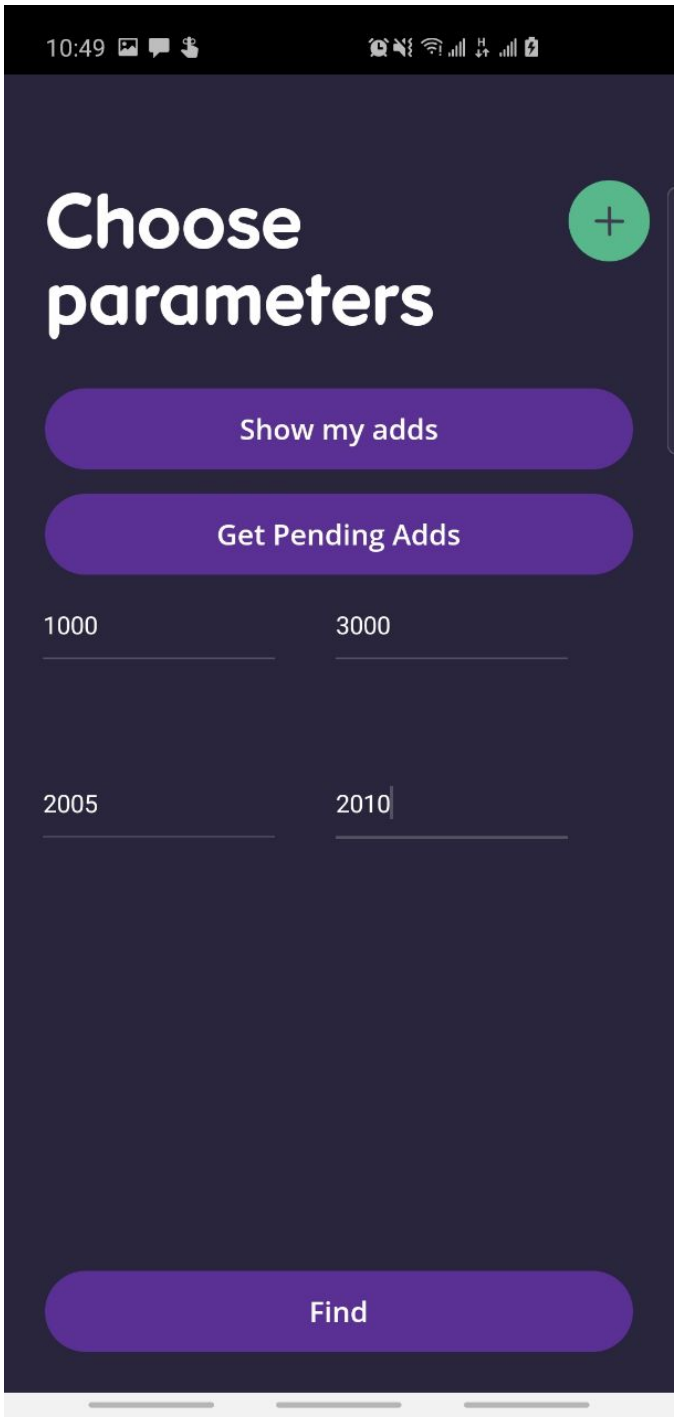

Рис. 4.12. Пошук оголошень. Фільтрація

Після натискання на кнопку пошуку, користувачеві буде показано екран зі списком доступних оголошень за вибраними ним параметрами.

4.7. Екран зі списком поданих оголошень

Рис. 4.13. Список оголошень

Даний екран має однаковий вигляд як для звичайного користувача, так і для модератора.

При виборі якогось з оголошень, користувача буде направлено на екран з деталями оголошення.

4.8. Екран з детальним описом оголошення

На екрані з деталями оголошення існує кілька шляхів взаємодії з сервісом, залежно від ролі користувача (модератор чи звичайний користувач), статусу оголошення (підтверджене модератором чи ні), а також від того, чи є поточний користувач автором поточного оголошення.

Модератор може видаляти будь-яке оголошення, а також підтверджувати оголошення, які він ще не перевірив.

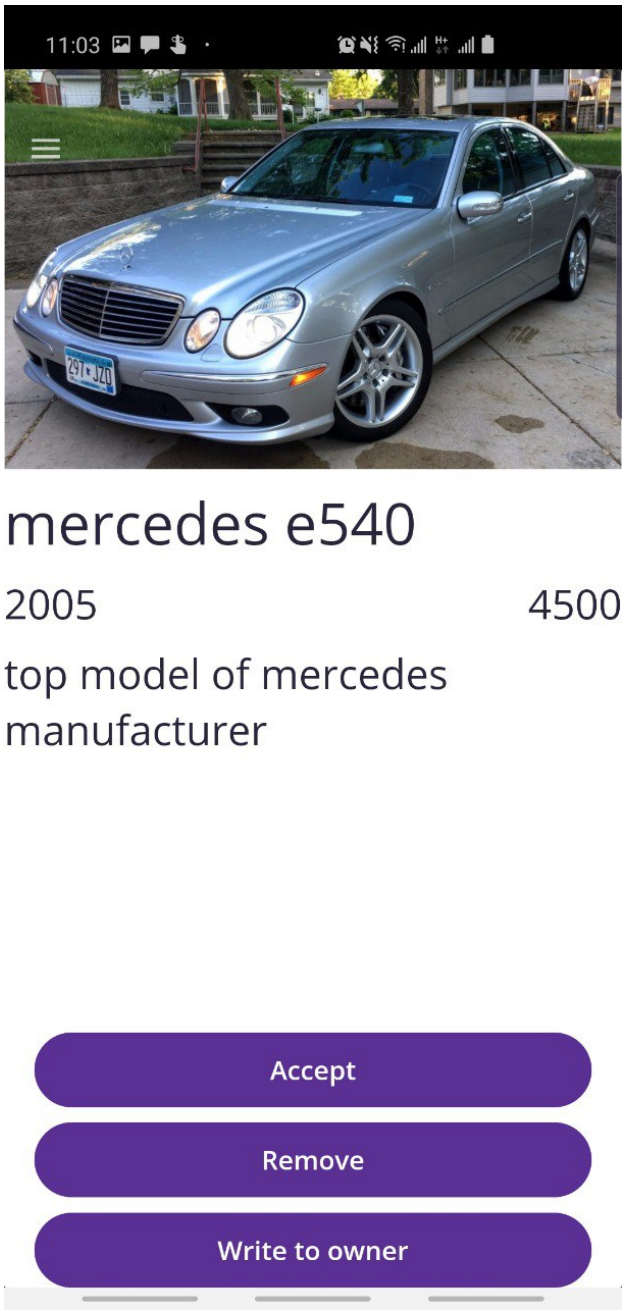

Рис. 4.14. Детальний опис оголошення адміністратора. Оголошення не підтверджене

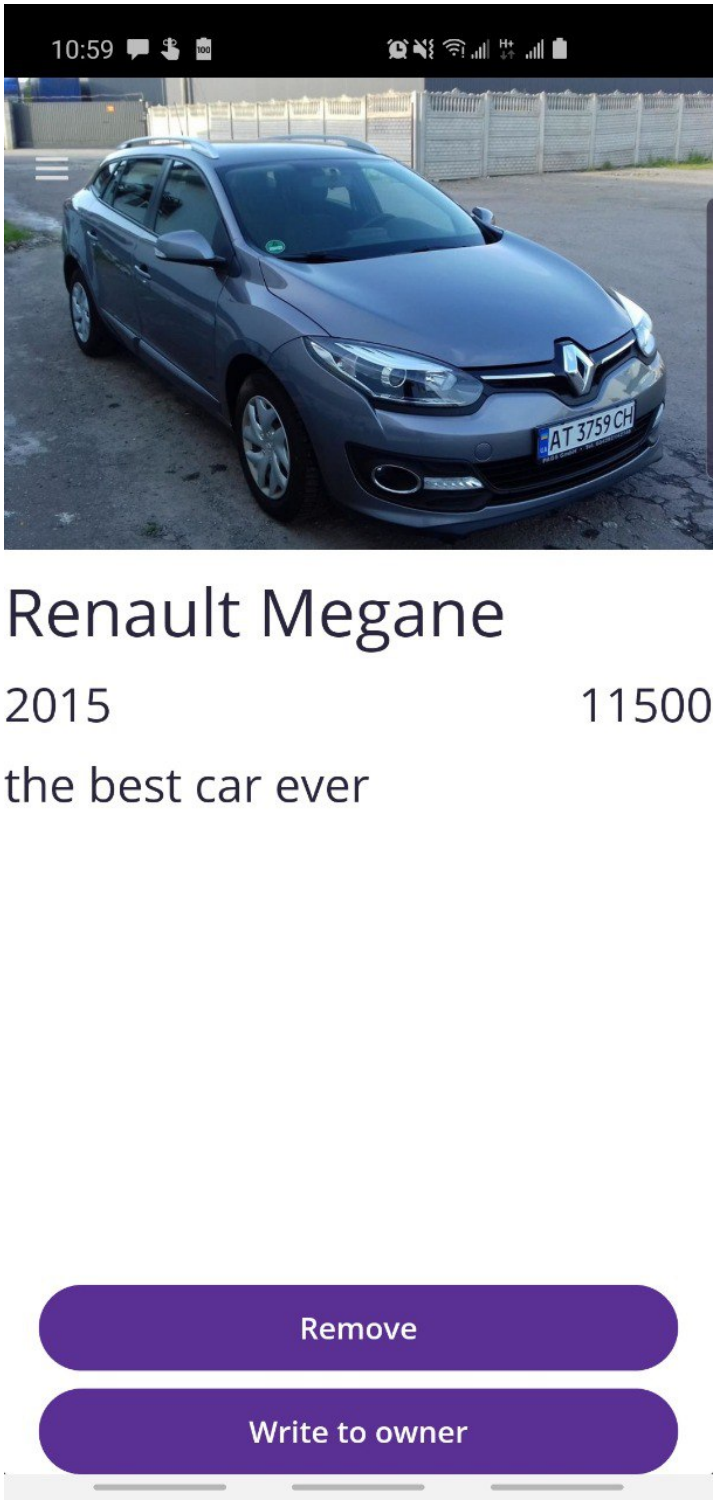

Рис. 4.15. Детальний опис оголошення адміністратора. Оголошення підтверджене.

Звичайний користувач може переглядати оголошення інших користувачів. У разі необхідності, він може зв’язатися з автором оголошення.

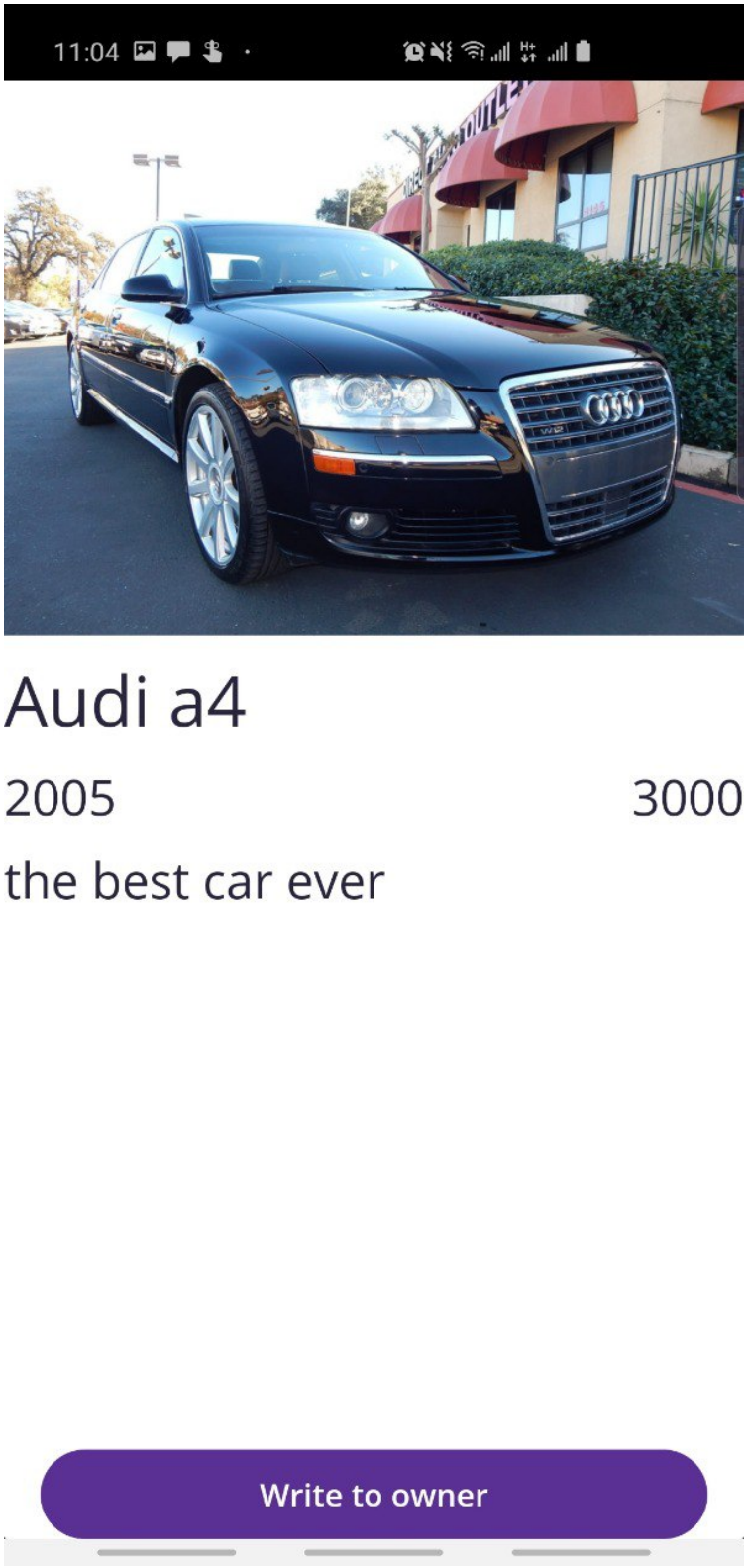

Рис. 4.16. Детальний опис оголошення

Користувач може зв’язатися з автором оголошення, натиснувши на кнопку “Write to owner”.

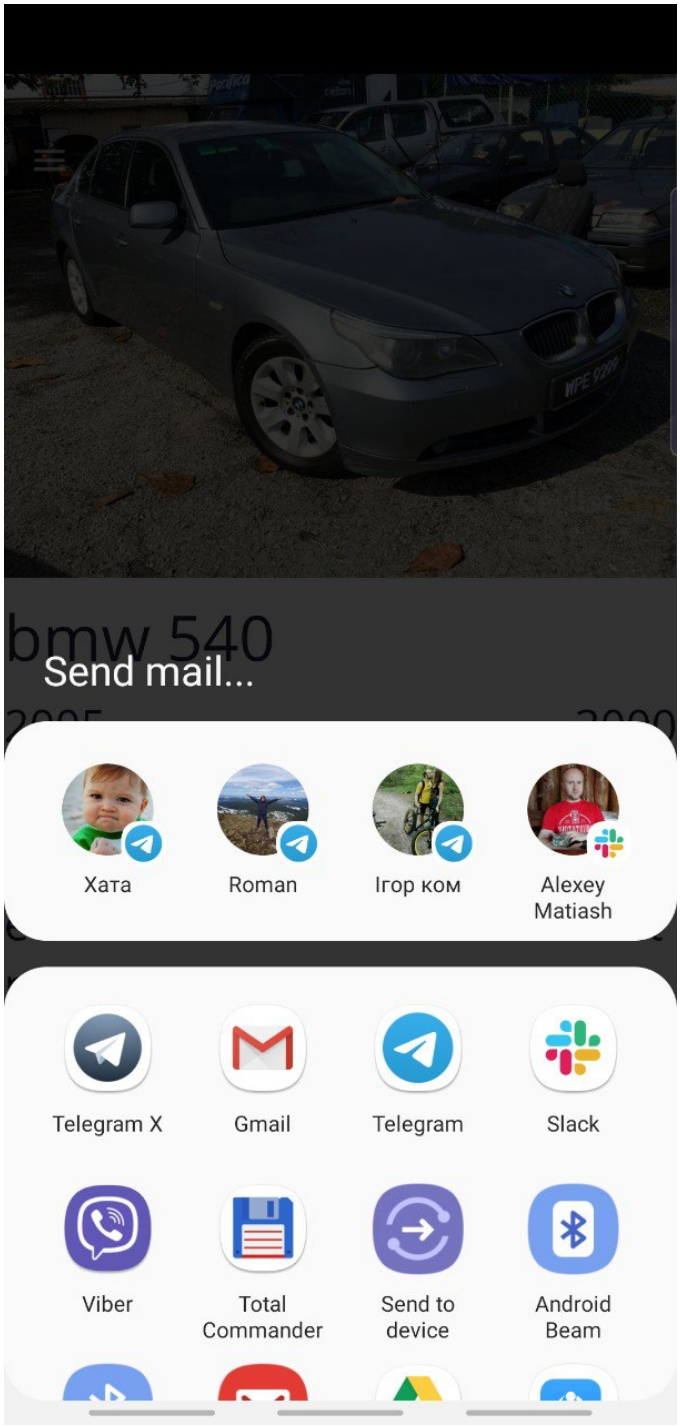

Рис. 4.17. Вибір можливості спілкування з автором оголошення

4.9. Екран створення оголошення

З екрану з вибором параметрів пошуку оголошень можна також перейти на екран додавання власного оголошення. Він має такий вигляд:

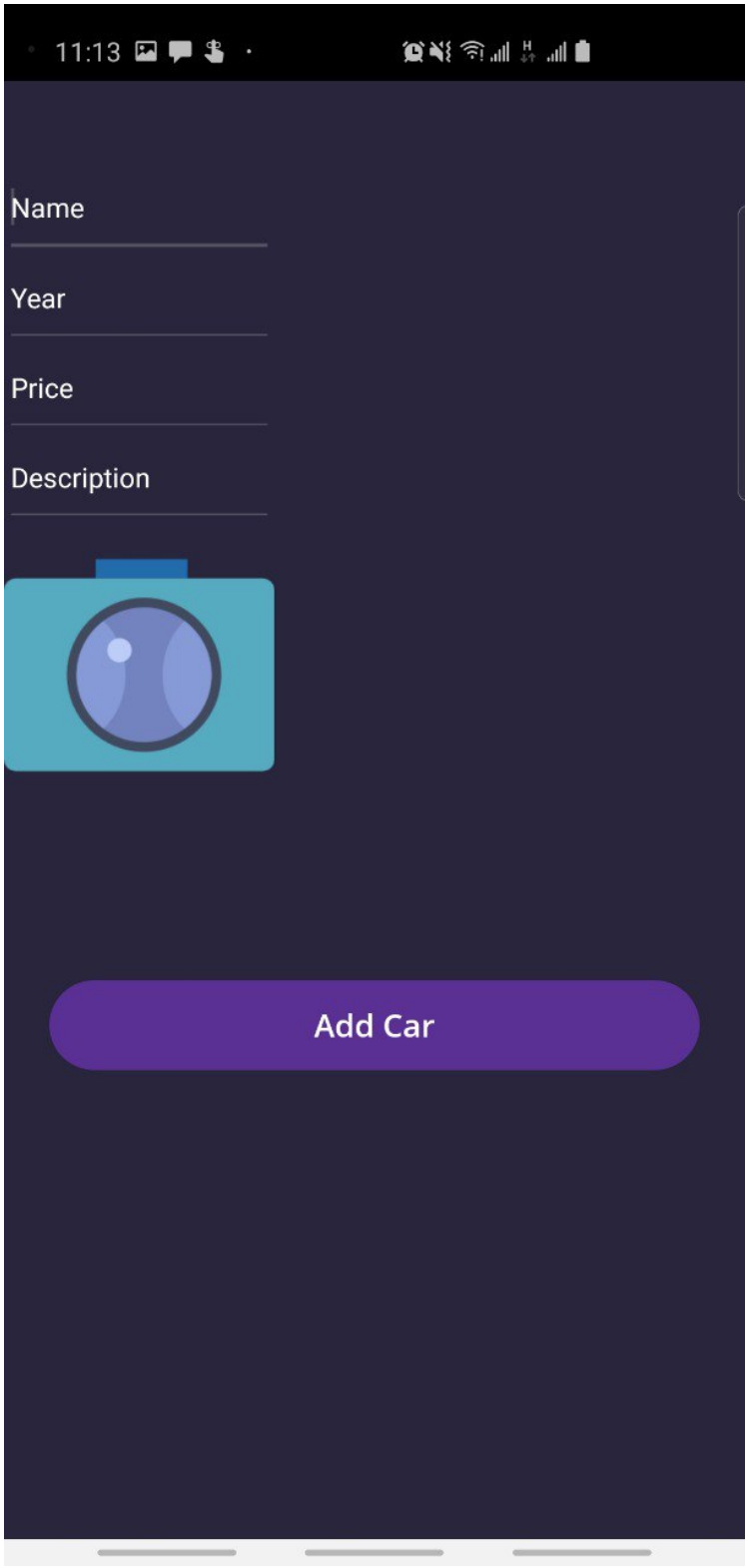

Рис. 4.18. Екран додавання оголошення

На даному екрані користувач повинен ввести необхідну інформацію про автомобіль, а також, за бажанням, додати фото авто.

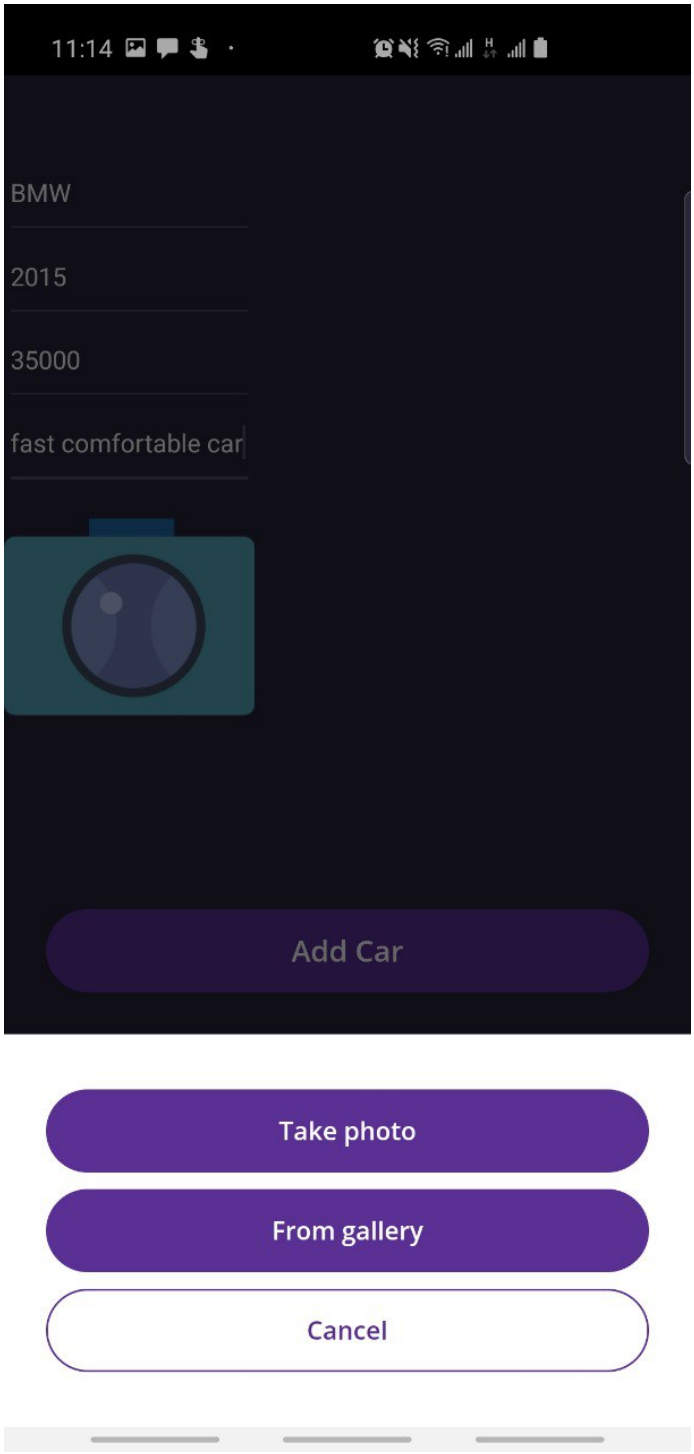

Рис. 4.19. Екран створення оголошення. Вибір фото транспортного засобу

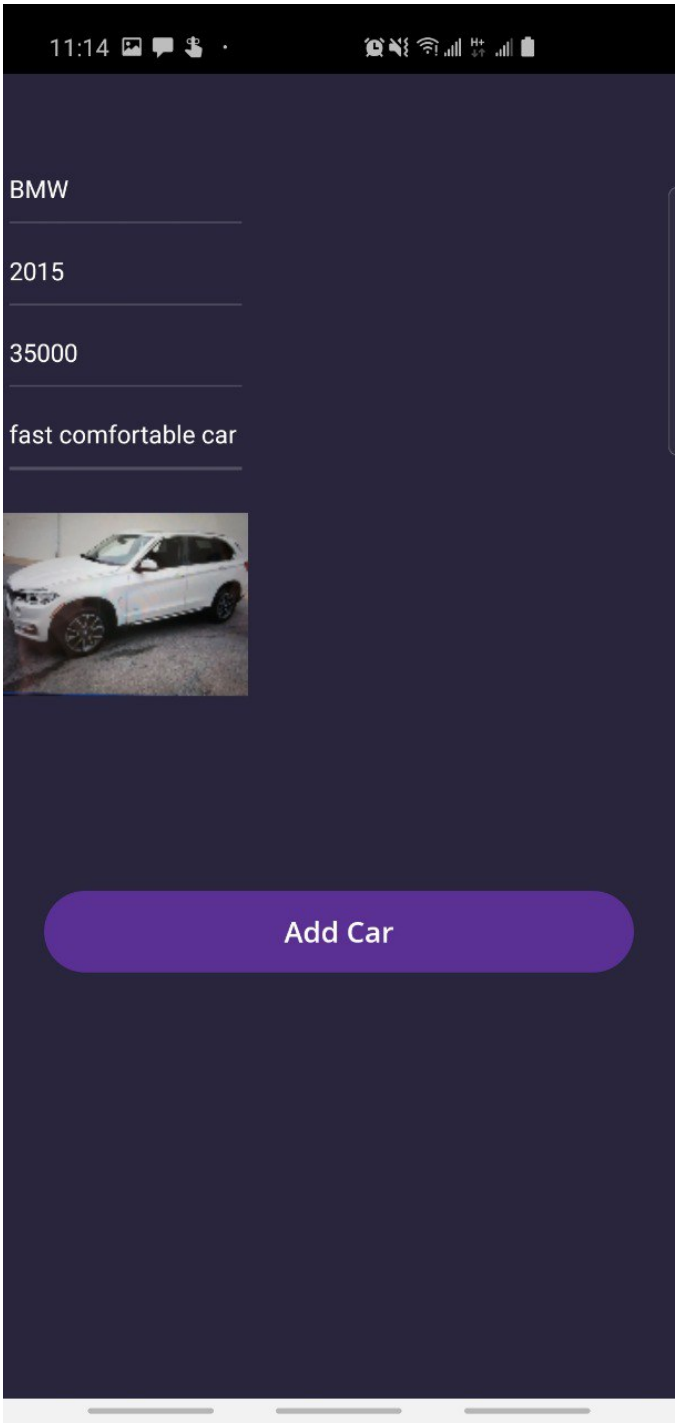

Рис. 4.20. Екран створення оголошення

4.10. Екран опцій користувача

На будь-якому екрані користувач може відкрити меню доступних йому опцій. Туди входять можливості додавання нового оголошення, а також виходу з системи.

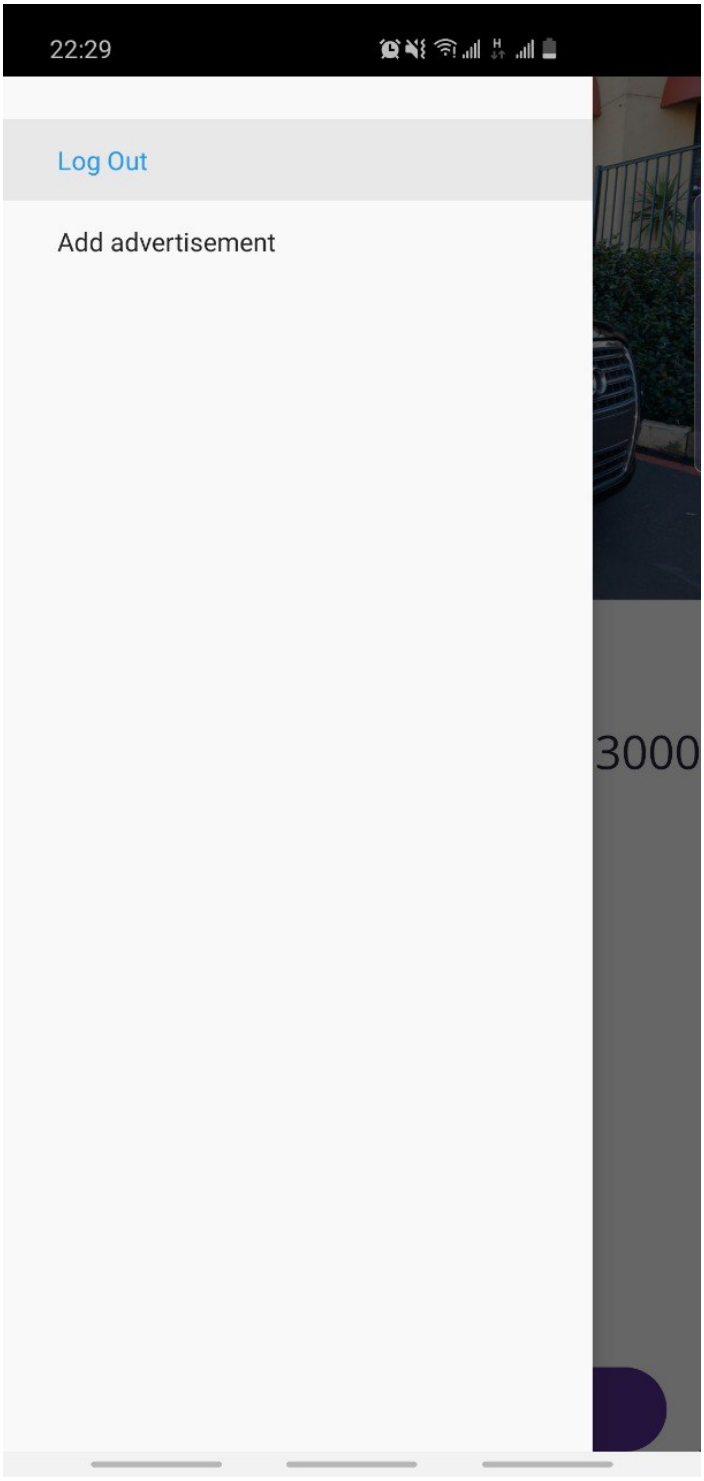

Рис. 4.21. Меню користувача

ВИСНОВОК ДО ЧЕТВЕРТОГО РОЗДІЛУ

Отже, в результаті роботи був створений android-клієнт сервісу для зручного пошуку оголошень про продаж автомобілів. Функціонал додатку включає в себе наступні можливості:

- Пошук всіх доступних автомобілів
- Реєстрація користувача
- Логін користувача
- Вибір автомобілів по фільтрам
- Завантаження власного авто на продаж
- Можливість видалення поданого оголошення
- Можливість перегляду нових оголошень про продаж авто модератором перед додаванням їх на основну дошку оголошень

РОЗДІЛ 5

РОЗРОБКА СТАРТАП-ПРОЕКТУ

Стартап — це тимчасова структура, яка займається пошуком рентабельної бізнес-моделі, що масштабується та відтворюється.

Основними складовими стартапу є [37]:

1. Ідея. Ідея – це початкова точка, з якої починається стартап. Вона має бути або зовсім свіжою і новою для обраного ринку, або включати в себе певні нововведення, що потенційно можуть стати конкурентною перевагою майбутнього продукту. Саме ідея перш за все має привабити потенційних інвесторів.
2. Команда. Команда стартапу зазвичай складається з кількох ентузіастів, які готові певний час працювати безкоштовно заради втілення ідеї. У кожного із членів команди має бути своя зона відповідальності, свій вклад в стартап і відповідно до вкладу – відсоток акцій.
3. Стартовий капітал. Зазвичай організатори стартапу намагаються залучити інвестиції для втілення своєї ідеї навіть на етапі розробки прототипу. Пізніше інвестори отримують свою частку дивідендів від продажу акцій стартапу. На ранніх етапах все ж можливі невеликі інвестиції з боку учасників стартапу, а не сторонніх інвесторів.

На етапі планування стартапу слід приділити увагу розробці інформаційної карти проекту, розподілу обов'язків між учасниками стартапу, визначення їх вкладу в проект, побудові морфологічної карти, побудувати бізнес модель, розробити ринкову стратегію проекту та маркетингову програму, проаналізувати ринкові можливості запуску стартап проекту, розробити виробничий та організаційний план. Усі ці кроки будуть реалізовані в підрозділах даного розділу.

5.1. Розробка інформаційної картки проекту

Інформаційна картка проекту проекту «Багатомодульна клієнтська платформа для трейдингу транспортними засобами» описана нище в таблиці 5.1.

Таблиця 5.1

Розробка стартап-проекту

1. Назва проекту	Віртуальний авто підбір
2. Автори проекту	Цехмейструк Роман
3. Коротка анотація	Розробка онлайн платформи (мобільний додаток), яка використовується для замовлень по підбору транспортних засобів у фахівців у цій області. Мобільний додаток має надавати змогу замовникам замовляти транспортні засоби з обраними характеристиками, а виконавцям – отримувати доступ до побажань користувача. Проект має бути масштабований. Алгоритм має легко адаптуватись до задач нових клієнтів. Система буде являти собою поєднання пошукової системи та надання покупцю рекомендацій на основі його попередніх звернень до системи. База даних побудована на основі вказаних раніше побажань. Наприклад, покупець обирає автомобіль, що відповідає вимогам – пробіг, потужність, клас авто і т.д. Після цього система рекомендує фахівців по підбору авто за заданими характеристиками. На будь якому етапі клієнт зможе повернутись на будь який попередній етап вибору та змінити параметри. Усі дані покупця зберігаються та накопичуються. Згодом, на основі закладених знань щодо вподобань даного покупця, актуальності класу авто, його пробігу та потужності тощо, система зможе пропонувати фахівців для підбору авто, які будуть відповідати вимогам.
4. Термін реалізації проекту	6 місяців *Тривалість проекту (в місяцях)*

Продовження таблиці 5.1

5. Необхідні ресурси	**Інтелектуальні**: Backend розробники – 2 чол (зп – 50 000 грн/міс), Android розробник – 1 чол (зп – 50 000 грн/міс), iOS розробник – 1 чол (зп – 50 000 грн/міс), QA – 1 чол (зп – 30 000 грн/міс), Системні адміністратори – 1 чол (зп – 30 000 грн/міс), UI/UX дизайнер – 1 чол (зп – 40 000 грн/міс), Product Owner – 1 чол (зп – 60 000 грн/міс), **Матеріальні**: Ноутбук MacBook Pro – 1 шт (ціна за шт - 55 000 грн), **Фінансові**: Ноутбук MacBook Pro – 7 шт (ціна за шт - 55 000 грн), Мобільні девайси для тестування (iPhone, Google Pixel, Samsung, iPad) – по 2 шт (ціна за шт – близько 30 000 грн), Ліцензія Figma повна – 1 шт (ціна 30 000 / рік), Ліцензія Amazon Services – 1 шт (ціна 500 000 / рік),, Сервіси Google Play/AppStore – по 1 шт (ціна 6 000 / рік),, Оренда коворкінга на 8 чол – 6 міс (ціна 20 000 / міс),
6. Опис проблеми, яку вирішує проект	Існує проблема підбору авто зі сторони клієнта з браком знань правильної оцінки транспортного засобу, це спонукає до вирішення проблеми з використанням кваліфікованих кадрів. Для підбору авто за заданими параметрами необхідно особисто переглядати багато варіантів, часто не маючи навіть базових знань у області оцінки технічного стану авто, а також відповідності авто до бажаних умов експлуатації. Як наслідок, клієнт має багато затраченого часу та часто обирає авто, яке не відповідає побажанням. Ідеєю додатку є надання послуг підбору авто від професіоналів у максимально короткі терміни.
7. Головні цілі та завдання проекту	**Цілі**: - Скоротити час на підбір авто, - Надати клієнту автомобіль, який повністю буде влаштовувати його побажання відповідно до затрачених коштів **Завдання**:

	- Створити систему для онлайн заповлень для підбору автомобіля за заданими параметрами і передача цих даних кваліфікованим спеціалістам

5.2. Визначення команди і розподіл задач і зон відповідальності

Засновнниками стартапу будуть 3 людини, ролі яких розподіляються наступним чином: генератор ідей (IT спеціаліст), спеціаліст (IT спеціаліст) та дипломат (юрист).

В таблиці 5.2 описаний орієнтовний графік реалізації проекту.

Таблиця 5.2

Орієнтовний графік реалізації проекту

1. Дослідження методів реалізації стартапу																			1,5m			
2. Видача ТЗ	1m													1m						1m		
3. Розподіл обов'язків														2,5m								
4. Розробка ПЗ														2m								
5. Тестування														1m								
6. Пошук інвесторів та покупців						4m																
7. Юредичне забезпечення					0,5m																	
8. Аналітика					0,5m																	
9. Рекламна компанія	2m																					

Розподілимо задачі між участниками стартапу. Задачі генератора ідей – розробка дизайну, проведення UX дослідження, участь в технічному дослідженні, тестування системи та проведення рекламної кампанії. В даному випадку генератор ідей виступає більше як product owner та бізнес аналітик, а також координує роботу рекламної агенції. Задачі спеціаліста – формування технічної команди, розробка програмного забезпечення та одночасно управління технічною командою. Тож в його обов'язки входить і технічне дослідження, і розробка, і тестування системи, і розробка інженерної схеми та монтаж тестового обладнання. Задачею дипломата є пошук інвесторів, аналітика, юридичне забезпечення, участь у формуванні рекламної стратегії та узгодження пропозицій рекламної агенції.

Нижче наведена схема взаємодії учасників стартапу.

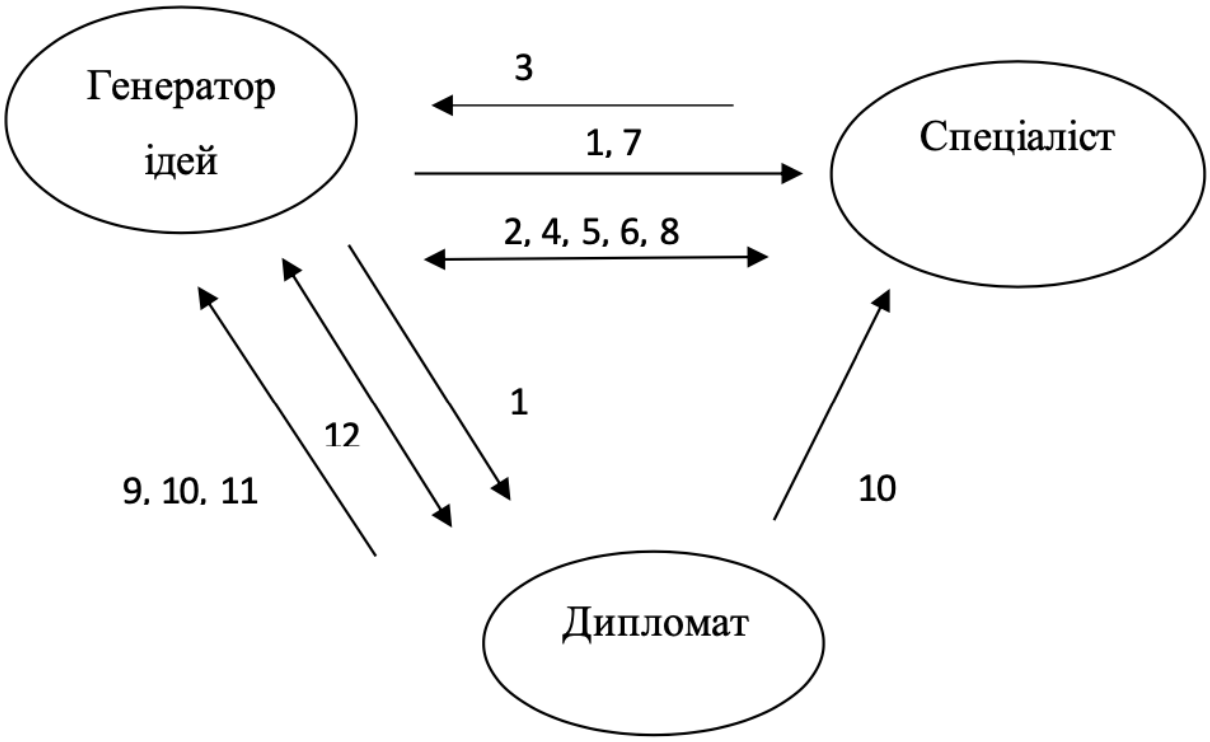

Рисунок 5.1. Схема взаємодії учасників проекту

Визначимо важливість вкладу кожного з учасників проекту. Ключові фактори оцінки ваги описані в таблиці 5.4.

Таблиця 5.4

Відносна вага факторів у реалізації проекту

Фактор	Вага
Ідея	4
Аналітика	5
Компетентність	6
Залученість і ризики	7
Обов'язки	6
Пошук інвесторів та покупців	8

Визначимо особистий внесок кожного з учасників проекту в проект.

Таблиця 5.5

Оцінка особистого внеску учасників стартапу

Фактор	Генератор ідей	Спецаіліст	Дипломат
Ідея	6	4	0
Аналітика	2	0	8
Компетентність	5	7	2
Залученість і ризики	2	2	2
Обов'язки	8	8	6
Пошук інвесторів та покупців	2	0	10

Наведемо зведену таблицю оцінки внеску учасників та ваги кожного з факторів.

Таблиця 5.6

Розподіл дольової участі партнерів у прибутку стартапу

Фактор	Вага	Генератор ідей	Спецаіліст	Дипломат	
Ідея	4	6x4=24	4x4=16	0	
Аналітика	5	2x5=10	0	8x5=40	
Компетентність	6	5x6=30	7x6=42	2x6=12	
Залученість і ризики	7	2x7=14	2x7=14	2x7=14	
Обов'язки	6	8x6=48	8x6=48	6x6=36	
Пошук інвесторів та покупців	8	2x8=16	0	10x8=80	
Разом		142	120	146	408
Дольовий відсоток у прибутку стартапу		35%	29%	36%	100%

5.3. Побудова морфологічної карти проекту

Визначення основних функцій

- Додаток має шифрувати дані користувачів для запобігання витоку даних;

- Додаток має бути візуально привабливим, зручним та зрозумілим для кінцевого користувача
- Платформа має бути масштабованою
- Платформа повинна мати адмінку для того, щоб клієнт міг самостійно змінювати питання/ певні параметри, завантажувати фото нових моделей одягу, видаляти застарілі моделі тощо
- Клієнт не повинен мати прямого доступу до серверу та БД. Для технічної підтримки платформи він має користуватись послугами стартап компанії
- Можливість отримання даних аналітики за період

Побудова морфологічної карти

Таблиця 5.7

Морфологічна карта

Основні параметри	Проміжні рішення				
	1-ше	2-ше	3-ше	4-ше	5-ше
Шифрування даних	Симетричні алгоритми (AES, DES)	Асиметричні алгоритми (El-Gamal, RSA)	Інше		
Дизайн	UI заснований на UX дослідженнях конкурентів	UI заснований на власних UX дослідженнях	Інше		

Масштабованість платформи (вибір технології роботи з БД)	Використання приватного блокчейн	Використання sql БД	Використання nosql БД	Інше	
Адмінка (вибір CMS)	Drupal	WordPress	Joomla	Bitrix	Django
Аналітика	Надання відкритого коду для підключення google аналітики, інтеграції CRM систем, тощо (клієнт самостійно або за допомогою сторонніх компаній формує звіти)	Самостійне (силами стартап компанії або підрядної компанії) формування звітів на основі витягу із БД та надання в період зазначений у договорі з клієнтом із наданням відкритого коду клієнту	Створення можливості автоматичного формування звітів на основі витягу із БД (по кліку кнопки в адмінці) без надання відкритого коду клієнту	Створення можливості автоматичного формування звітів на основі витягу із БД (по кліку кнопки в адмінці) із наданням відкритого коду клієнту	Інше

Товар за задумом: система для онлайн підборів авто за заданими параметрами через кваліфікованих спеціалістів

Товар за реалізацією: систему для онлайн заповлень для підбору автомобіля за заданими параметрами і передача цих даних кваліфікованим спеціалістам

Товар з підкріпленням: система з готовим набором бажаних характеристик автомобіля відповідно до групи користувачів системи

5.4. Побудова бізнес моделі стартапу та розробка ринкової стратегії проекту

Таблиця 5.6

Вибір цільових груп потенційних споживачів

№ п/п	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1	Авто площадки, фізичні особи, які є кваліфікованими спеціалістами в області підбору автомобілів	Висока, через брак конкурентів у обраній сфері продажу-купівлі авто	Високий, прямих аналогів немає	Низька – через відсутність систем зі схожим функціоналом	Висока через високу готовність входу і низьку інтенсивність
2	Авто магазини та автосалони	Висока, через брак конкурентів у обраній сфері продажу-купівлі авто	Високий, прямих аналогів немає	Середня – через актуальність технологій, які використовуються топовими брендами	Висока через високу готовність входу і низьку інтенсивність
3	Юридичні особи, які займаються продажем авто	Низька, через брак конкурентів у обраній сфері продажу-купівлі авто	Високий, прямих аналогів немає	Висока через пряму схему взаємодії згаданих осіб з клієнтами	Низька через низьку готовність входу і високу інтенсивність

Таблиця 5.7

Визначення базової стратегії розвитку

№ п/п	*Обрана альтернатива розвитку проекту*	*Стратегія охоплення ринку*	*Ключові конкурентоспроможні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку**
	Переорієнтація проекту в бік **кінцевого споживача.** Додаток пропонує скласти «портфель» із бажаних характеристик автомобіля	Стратегія концентрованого ринку (конкретний сегмент споживачів)	Відсутність аналогів, user-friendly інтерфейс	Стратегія диференціації

Таблиця 5.8

Визначення базової стратегії конкурентної поведінки

№ п/п	*Чи є проект «першопрохідцем» на ринку?*	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки**
	Так	І те і інше	Ні, немає прямих конкурентів	Стратегія заняття конкурентної ніші

Таблиця 5.9

Визначення стратегії позиціонування

№ п/п	*Вимоги до товару цільової аудиторії*	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап-проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)*
	Зручність, UI/UX Отримання розширених унікальних даних для аналітики	Стратегія диференціації	- Відсутність аналогів, - Етичний збір унікальних даних споживачів, - Продукт має цінність і для покупця (автоплощадка) і для кінцевого користувача (клієнту)	Швидкість підбору, Відповідність автомобіля бажанням клієнта, Кількість запропонованих варіантів зі сторони автоплощадки

	Легкість формування аналітичних звітів Масштабованість Надійність		- Продукт є не тільки аналітичним, але й створює велику лояльність серед покупців авто автоплощадок	

5.5. Аналіз ринкових можливостей стартап проекту

Для оцінки ринкових можливостей запуску стартап проекту та визначення своєчасності його створення, слід проаналізувати рентабельність ринку, фактори загроз і можливостей, визначити фактори конкурентосмроможності. Тільки на основі цих даних можливо прийняти рішення про те слід запускати стартап в його поточному вигляді, слід почекати, або слід модифікувати ідею, обрати іншу ринкову стратегію тощо [38].

Спочатку складемо попередню характеристику потенційного ринку проекту.

Таблиця 5.10

Попередня характеристика потенційного ринку стартап-проекту

п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	Безліч
2	Загальний обсяг продаж, грн/ум.од	-
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Отримання патенту на розроблений метод управління системою «Розумний дім»
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	40%

Як вже згадувалось раніше, ринок конкурентний, однак при цьому він постійно зростає, що при відсутності особливих обмежень, дає можливість зайняти свою нішу.

Таблиця 5.11

Фактори загроз виходу стартап-проекту на ринок

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Копіювання конкурентами	Конкуренти можуть скопіювати технологію та створити аналогічний продукт (можливо більш дешевий або кращої якості	Патентування розробленого методу управління системою «Розумний дім»
2	Пандемія	Пандемія негативно вплинула на доходи більшості домогосподарств	Донесення до споживача інформації про те як система дозволить заощадити в майбутньому за рахунок економії електроенергії, тощо. Донесення до споживача інформації про те, що в умовах самоізоляції комфортне житло та автоматизація домашніх процесів набуває набагато більшого значення ніж раніше
3	Невідома стартап компанія	Недовіра з боку споживачів	Залучитись рекомендаціями з боку лідерів думок та професіоналів

Таблиця 5.12

Фактори можливостей виходу стартап-проекту на ринок

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Пандемія	В умовах вимушеної самоізоляції комфорт вдома став надзвичайно важливим і люди готові витрачати більше коштів на девайси та послуги, які зроблять самоізоляцію більш приємною	Закладення можливості швидкого розширення та пришвидшення виходу на ринок
2	Розвиток НТП	Можуть з'явитись нові технології які дозволять скоротити бюджети/модернізувати систему, переорієнтувати її тощо	Готовність оперативно користуватися наданими можливостями
3	Цікавість з боку інших ринків	Споживачі на інших ринках можуть зацікавитись ідеєю та замовити адаптовану систему	Створити альтернативний бізнес план та стратегію розвитку

Проаналізувавши фактори загроз та можливостей, можна зробити висновок що пандемія є не стільки ризиком, скільки чудовою можливістю для стартапу.

Люди, які бажають придбати автомобіль, з більшою долею імовірності будуть шукати авто онлайн.

Таблиця 5.13

Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
Тип конкуренції – чиста конкуренція	Безліч компаній пропонують схожі за функціоналом системи управління	Робота над розширенням можливостей та покращенням технічних характеристик продукту
За рівнем конкурентної боротьби – міжнародне конкурентне середовище	Ринок наповнений закордонними аналогами та товарами замінниками	Приділення уваги локалізації, можливість виходу на міжнародний ринок
За галузевою ознакою – міжгалузева	Система автоматизації потенційно може використовуватись в інших сферах	Можливість вийти на інші ринки із адаптованою системою
Конкуренція за видами товарів – товарно-видова	Існують різні технології побудови та управління системою	Розширення функціоналу, підвищення рівня безпеки
За характером конкурентних переваг – цінова конкуренція	Є безліч подібних систем, та розумних пристроїв зі своїм управлінням.	Необхідно зробити систему не тільки зрозумілою для користувача, але й конкурентною за ціною.

Аналізуючи всі отримані вище дані ступеневого аналізу, необхідно провести аналіз конкуренції в галузі за Порнетом.

Таблиця 5.14

Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	В Україні існує кілька десятків компаній, що надають повний спектр послуг по проектуванню та підключення системи під ключ, а також онлайн платформи, що продають схожі рішення	Вітчизняні та іноземні компанії, що займаються розробкою ПЗ	Виробники мікроконтролерів та розумних пристроїв	Клієнти мають обмежений бюджет. Клієнти ставляться до систем «Розумний дім» як до чогось мало доступного і незвичного	Як таких замінників немає

Аналізуючи ці дані, бачимо, що найефективнішим варіантом розвитку подій в разі недостачі бюджету для реклами є співпраця напряму з автоплощадками, які готові продовити торги транспортними засобами через платформу, яка розробляється.

Таблиця 5.15

SWOT-аналіз стартап проекту

Сильні сторони: Ціна Функціональність системи Інтерфейс	Слабкі сторони:
Можливості: Пандемія Цікавість з боку інших ринків Розвиток НТП	Загрози: Копіювання конкурентами Пандемія та зниження доходів споживачів Поява нових конкурентів

На основі SWOT-аналізу розробимо також альтернативну стратегію виходу проекту на ринок.

Таблиця 5.16

Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Залучення додаткових інвестицій	Середня	3-4 місяці
2	Впровадження акцій, відтерменувань платежів	Висока	1-2 місяці
3	Продаж через забудовників, дизайнерські та архітектурні бюро. В т.ч. включення системи в ремонт «під ключ» від забудовника.	Висока	2-3 місяці

5.6. Розробка виробничого плану та розрахунок витрат для запуску проекту

Скласти календарний план-графік реалізації проекту за формою, наведеною в табл. 5.17.

Таблиця 5.17

Календарний план-графік реалізації стартап-проекту

№ з/п	*Етапи реалізації*	*Період реалізації проекту*						
		0-й рік[1]				*1-й рік*	*2-й рік*	*3-й рік*
		1-й кв.	*2-й кв.*	*3-й кв.*	*4-й кв.*			
1.	Проведення НДДКР	+						
2.	Розробка проектних матеріалів і ТЕО		+					
3.	Робоче проектування і прив'язка проекту			+				
4.	Створення компанії		+					
5.	Придбання нематеріальних активів, отримання дозвільних документів тощо		+					
6.	придбання й оренда земельних ділянок, будівель, приміщень, споруд				+			
7.	Придбання обладнання, устаткування та пристроїв				+			
8.	Передвиробничі маркетингові дослідження			+				
9.	Приймально-здавальні випробування				+			
10.	Пусконалагоджувальні роботи				+			
11.	Освоєння проектних потужностей				+			
13.	Придбання матеріальних ресурсів				+			

Визначити потребу у основних засобах (земельних ділянках, будівлях, приміщеннях, спорудах, передавальних пристроях, обладнанні), необхідних для реалізації проекту, та умови їх використання за формою, наведеною в табл. 5.17.-5.18.

Таблиця 5.18

Планова потреба у виробничих площах

№ з/п	*Тип приміщення (будівлі, ділянки, споруди)*	*Кількість одиниць*	*Площа, кв. м*	*Вимоги до приміщення (будівлі, ділянки, споруди)*	*Умови надання*	*Вартість, грн.*
1.	Коворкінг	1	50	Окремий опен спейс із меблями, кухнею та переговорною зоною	оренда	20000 грн/міс
Разом (рік)				—	—	240000

Площа необхідних приміщень визначається на підставі потреб у [i]:

- *виробничому приміщенні*, площа якого ($S_{пр}$) може бути визначена за формулою:

$$S_{пр} = \sum_{i=1}^{m} S_i \cdot O_i \cdot k_f ,$$

де m – кількість операцій технологічного процесу виготовлення виробів;

S_i – габарити обладнання для виконання i-й операції, кв. м;

O_i – кількість обладнання для виконання i-й операції, одиниць;

k_f – коефіцієнт, що враховує потребу у додатковій площі (k_f = 2,0-3,0);

- *складському приміщенні,* площа якого може бути наближено визначена в розмірі 30-50% від площі виробничого приміщення;
- *офісному приміщенні,* площа якого може бути прийнята в межах 20-30 кв. м.

Таблиця 5.18

Планова потреба у виробничому обладнанні та устаткуванні

№ з/п	*Вид обладнання (устаткування, пристрою)*	*Терміни постачання*	*Вартість, грн.*
2.	Сервери Amazon	1 міс	100 000грн/міс
3.	Комп'ютери MacBook Pro, 8 шт	10 днів	400 000 грн
7.	Мобільні девайси iPhone X, Samsung Gallaxy Note, iPad	3 дні	160 000грн
Разом:		—	*660 000 грн*

Вартість обладнання може бути визначена наступним чином [2]:

– *вартість технологічного обладнання* – за формулою:

$$K_{mo} = \sum_{i=1}^{m} O_i \cdot Ц_i,$$

де m – кількість операцій технологічного процесу виготовлення продукції;

O_i – кількість одиниць обладнання для виконання *і-й* операції;

$Ц_i$ – ціна одиниці обладнання для виконання *і*-й операції.

– *вартість допоміжного обладнання* може бути визначена наближено на рівні приблизно 30% від вартості технологічного обладнання;
– *вартість інвентарю* також може бути визначена наближено на рівні 10-15% від вартості технологічного обладнання.

Визначити обсяг витрат на залучення нематеріальних активів, необхідних для реалізації стартап-проекту за формою, наведеною в табл. 5.4.

Таблиця 5.19

Планова вартість нематеріальних активів

№ з/п	*Вид активів*	*Активи, що можуть бути віднесені до даного виду*	*Вартість, грн.*
1.	Ліцензії на програмні продукти	Adobe, Atlassian, etc.	90 000

Визначимо плановий обсяг виробництва продукції стартап-проекту (в натуральних показниках) по роках за формою, наведеною в табл. 5.5.

Таблиця 5.20

Плановий обсяг виробництва продукції стартап-проекту

Вид продукції	*Одиниця виміру*	*Обсяги виробництва за період*		
		1-й рік	*2-й рік*	*3-й рік*
Онлайн платформа	шт	1	1	1
Мобільний додаток iOS	шт	1	1	1
Мобільний додаток на Android	шт	1	1	1

Визначимо обсяг витрат на забезпечення стратап-проекту матеріальними ресурсами та комплектуючими по роках (виходячи з планового обсягу виробництва продукції, визначеного в табл. 5.5) за формою, наведеною в табл. 5.2. Якщо проектом передбачено виробництва декількох видів продукції, таблиця складається по кожному виду продукції окремо.

Таблиця 5.21

Планова потреба у матеріальних ресурсах та комплектуючих

№ з/п	*Вид ресурсу*	*Одиниця виміру*	*Витрати на одиницю продукції в натуральних показниках*	*Вартість на одиницю продукції, тис. грн.*	*Вартість за плановим обсягом виробництва за період, тис грн.*		
					1-й рік	*2-й рік*	*3-й рік*
1.	Матеріали						
1.1.							
…							
Всього матеріалів		—	—				
2.	Комплектуючі						
2.1.							
…							
Всього комплектуючих		—	—				
3.	Сировина						
3.1.							
…							
Всього сировини							
Разом:		—	—				

Визначимо потребу та обсяг витрат на залучення адміністративного та промислово-виробничого персоналу, необхідного для реалізації проекту за формою, наведеною в табл. 5.17.

Таблиця 5.22

Планова потреба та витрати на персонал

№ з/п	*Категорія персоналу*	*Чисель-ність*	*Заробітна плата, грн. на місяць*	*Відрахування на соціальні заходи, тис грн. на місяць*	*Витрати на оплату праці за період, тис. грн.*		
					1-й рік	*2-й рік*	*3-й рік*
1.	Адміністративний персонал (працівники апарату управління)						
1.1.	Фінансовий директор	1	20000	6600	600	600	600
1.4.	HR	1	16000	5000	300	300	300
Всього витрати на оплату праці адміністративного персоналу							
2.	Промислово-виробничий персонал						
2.1.	Технічний директор	1	20000	6600	1200	1200	1200
2.2.	Виконавчий директор	1	20000	6600	1200	1200	1200
2.7	Системний адміністратор	1	10000	3400	1200	1200	1200

№ з/п	*Категорія персоналу*	*Чисель-ність*	*Заробітна плата, грн. на місяць*	*Відрахування на соціальні заходи, тис грн. на місяць*	*Витрати на оплату праці за період, тис. грн.*		
					1-й рік	*2-й рік*	*3-й рік*
2.8	QA manual	4	16000	5500	1632	1632	1632
2.8	Frontend розробники (Android + iOS)	2	18000	6300	2400	2400	2400
2.9	Backend розробники	2	18000	6300	2400	2400	2400
Всього витрати на оплату праці промислово-виробни-чого персоналу							
	Разом:				19032	19032	19032

З урахуванням даних табл. 5.14-5.21 визначимо обсяг загальних початкових витрат, необхідних для реалізації проекту (витрат, що мають бути понесені до початку основної діяльності в 0-й рік реалізації проекту) за формою, наведеною в табл. 5.23.

Таблиця 5.23

Загальні початкові витрати проекту

№ з/п	*Стаття витрат*	*Обсяги витрат в 0-й рік, тис. грн.*
1.	Проведення НДДКР	300
2.	Розробка проектних матеріалів і ТЕО	0

№ з/п	*Стаття витрат*	*Обсяги витрат в 0-й рік, тис. грн.*
3.	Робоче проектування і прив'язка проекту	0
4.	Витрати на придбання й оренду земельних ділянок, будівель, приміщень, споруд	240
5.	Витрати на придбання обладнання та устаткування та пристроїв	2 326
6.	Витрати на приймально-здавальні випробування	0
7.	Витрати на пусконалагоджувальні роботи	0
8.	Комплексне освоєння проектних потужностей	0
9.	Витрати на придбання нематеріальних активів	100
10.	Одноразові виплати, зокрема гарантуючим і страховим організаціям	0
11.	Витрати на створення оборотного капіталу, необхідного для початку операційної діяльності (створення виробничих запасів, передоплата сировини, матеріалів і комплектуючих виробів, які мають бути поставлені на початку операційної діяльності)	0
12.	Податкові платежі (земельний, комунальний та інші), здійснені до початку операційної діяльності	0
13.	Оплата юридичних послуг	100
14.	Витрати на передвиробничі маркетингові дослідження і створення збутової мережі	200
15.	Витрати, пов'язані з діяльністю персоналу	19032
16.		
	Разом	*22 298*

Визначимо обсяг поточних загальногосподарських витрат, необхідних для реалізації проекту, за формою, наведеною в табл. 5.23.

Таблиця 5.24

Планові загальногосподарські витрати

№ з/п	*Стаття витрат*	*Витрати за період, тис. грн.*		
		1-й рік	*2-й рік*	*3-й рік*
1.	Витрати на оренду земельних ділянок, будівель, приміщень, споруд	240	240	240
2.	Витрати на обладнання, устаткування та пристрої	0	0	0
3.	Витрати на придбання нематеріальних активів	100	100	100
4.	Витрати на персонал (на відрядження, соціальні заходи тощо)	19032	19032	19032
5.	Витрати на зв'язок	0	0	0
6.	Витрати на паливо та електроенергію	0	0	0
7.	Витрати на водопостачання	0	0	0
8.	Витрати на утримання обладнання та приміщень	0	0	0
9.	Витрати на збут	0	0	0
10.	Витрати на просування та рекламу	500	700	1000
11.	Оплата юридичних послуг	100	50	50
12.	Податкові платежі (земельний, комунальний податки, інші)	0	0	0
…				
	Разом:	19972	20172	20472

Отже, загальна ціна проекту – 20472 грн.

ВИСНОВОК ДО П'ЯТОГО РОЗДІЛУ

В даному розділі був проведений аналіз розробки стартап проекту «Багатомодульна клієнтська платформа для трейдингу транспортними засобами». Були визначені переваги і недоліки стартапу, описано стратегію і альтернативну стратегію для виводу проекту на ринок. Був описаний календар проведення робіт.

Цілевою аудиторією було визначено клієнтів, яким необхідно знайти автомобіль у найбільш короткий термін, не витрачаючи на це час або ресурси.

Конкурентними перевагами системи є її зручність, багатофункціональність та приваблива ціна. Саме останній пункт дозволяє включити в сегмент споживачів середнього плюс класу в цільову аудиторію стартапу.

Було прораховано плановий термін реалізації першої версії системи на основі розробленого прототипу, а також розраховано суму інвестицій для виходу на ринок.

ВИСНОВКИ

В результаті роботи над даною дипломною роботою був реалізований програмний додаток, який є клієнтською частиною багатомодульної платформи для торгівлі транспортними засобами.

Код програмного додатку був написаний відповідно до сучасних найкращих практик написання програмного коду під операційну систему Android.

Середовищем розробки була обрана IDE Android Studio.

Як результат роботи отримано файл збірки apk, який готовий до встановлення на девайси, які працюють під операційною системою Android. Цей файл може бути розповсюдженим через систему Google Play.

Спілкування клієнтської частини з бекендом відбувається через REST Api.

Ключовою відмінністю даного проекту від більшості робіт-аналогів є те, що дана робота була реалізована, використовуючи багатомодульний підхід.

В результаті роботи був створений стартап проект для виходу на ринок з готовим продуктом.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] Офіційна документація Spring [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: https://spring.io/docs.

[2] Spring Recipes: A Problem-Solution Approach – London: Apress, 2012. – 2312 с.

[3] MySQL Database [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://dev.mysql.com/doc/.

[4] Eureka Server [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://cloud.spring.io/spring-cloud-netflix/multi/multi_spring-cloud-eureka-server.html.

[5] Офіційна документація Spring Cloud [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://spring.io/projects/spring-cloud.

[6] Офіційна документація Apache Kafka [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: https://kafka.apache.org/documentation/.

[7] Open Feign [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://spring.io/projects/spring-cloud-openfeign.

[8] Офіційна документація Docker [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://docs.docker.com/.

[9] Autobahn-tech [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: http://autobahn.tech/.

[10] Wizzle [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.wizzle.co.uk/.

[11] Carwago [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://carvago.com/.

[12] ADESA [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.adesa.eu/.

[13] Zig Wheels [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.zigwheels.com/.

[14] SwapMotors [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.swapmotors.com/.

[15] Car Wow [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.carwow.co.uk/.

[16] Car Some [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://www.carsome.my/.

[17] Microservice architecture patterns [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://microservices.io/patterns/microservices.html/.

[18] Офіційна документація Hibernate [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://hibernate.org/orm/documentation/5.4/.

[19] Amazon Web Services [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://aws.amazon.com/.

[20] Служба хмарних обчислень Azure [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://azure.microsoft.com/.

[21] Офіційна документація EclipseLink [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://www.eclipse.org/eclipselink/.

[22] AMPQ Overview [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://www.amqp.org/.

[23] Офіційна документація Zuul [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://zuul-ci.org/docs/zuul/.

[24] Zuul API [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: http://netflix.github.io/zuul/javadoc/zuul-core/index.html?help-doc.html /.

[25] Spring Cloud Projects [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://projects.spring.io/spring-cloud/spring-cloud.html/.

ДОДАТКИ

ДОДАТОК А

Лістинг програмного коду

```
public class AddCarPresenter extends BasePresenter<AddCarView> {

    private final ICarsDataSource mCarsDataSource;

    public AddCarPresenter(ICarsDataSource carsDataSource) {
        mCarsDataSource = carsDataSource;
    }

    public void addCar(String header, CreateCarDto createCarDto) {
        AddCarView view = getView();

        addSubscription(
                mCarsDataSource
                        .addCar(header, createCarDto)
                        .retryWhen(new RxRetryWithDelay())
                        .subscribeOn(Schedulers.io())
                        .observeOn(AndroidSchedulers.mainThread())
                        .subscribe((t) -> view.onAdded(t), new RxErrorAction(view)));
    }

    public void uploadCar(String header, Integer carId, String filePath) {
        AddCarView view = getView();

        addSubscription(
                mCarsDataSource
                        .uploadImage(header, carId, filePath)
                        .retryWhen(new RxRetryWithDelay())
                        .subscribeOn(Schedulers.io())
                        .observeOn(AndroidSchedulers.mainThread())
                        .subscribe((t) -> Log.i("IMAGE", "image uploaded"), new
RxErrorAction(view)));
    }
}
public class BasePresenter<T extends BaseView> extends MvpBasePresenter<T>
        implements MvpPresenter<T> {

    private CompositeSubscription mCompositeSubscription = new CompositeSubscription();

    public void destroy() {
        mCompositeSubscription.clear();
    }

    protected Subscription addSubscription(Subscription subscription) {
        mCompositeSubscription.add(subscription);
        return subscription;
    }
}
public class CarListPresenter extends BasePresenter<CarListView> {

    private final ICarsDataSource mCarsDataSource;

    public CarListPresenter(ICarsDataSource carsDataSource) {
        mCarsDataSource = carsDataSource;
    }
```

```
    public void getCars(String header, Integer yearFrom, Integer yearTo, Integer priceFrom,
Integer priceTo) {
        CarListView view = getView();

        addSubscription(
                mCarsDataSource
                        .getCars(header, yearFrom, yearTo, priceFrom, priceTo)
                        .retryWhen(new RxRetryWithDelay())
                        .subscribeOn(Schedulers.io())
                        .observeOn(AndroidSchedulers.mainThread())
                        .subscribe(view::showCarList, new RxErrorAction(view)));
    }

    public void getMyCars(String header, String id) {
        CarListView view = getView();

        addSubscription(
                mCarsDataSource
                        .getUserCars(header, id)
                        .retryWhen(new RxRetryWithDelay())
                        .subscribeOn(Schedulers.io())
                        .observeOn(AndroidSchedulers.mainThread())
                        .subscribe(view::showCarList, new RxErrorAction(view)));
    }

    public void getPendings(String header) {
        CarListView view = getView();

        addSubscription(
                mCarsDataSource
                        .getPendings(header)
                        .retryWhen(new RxRetryWithDelay())
                        .subscribeOn(Schedulers.io())
                        .observeOn(AndroidSchedulers.mainThread())
                        .subscribe(view::showCarList, new RxErrorAction(view)));
    }
}

public class ForgotPasswordPresenter extends BasePresenter<ForgotPasswordView> {

    private final IAccountDataSource mAccountDataSource;

    public ForgotPasswordPresenter(IAccountDataSource accountDataSource) {
        mAccountDataSource = accountDataSource;
    }

    public void onForgotPassword(EmailDto emailDto) {
        ForgotPasswordView view = getView();

        addSubscription(
                mAccountDataSource
                        .forgotPassword(emailDto)
                        .retryWhen(new RxRetryWithDelay())
                        .subscribeOn(Schedulers.io())
                        .observeOn(AndroidSchedulers.mainThread())
                        .subscribe(view::showSuccessDialog, new RxErrorAction(view)));
    }
}
public class FullCarInfoPresenter extends BasePresenter<FullCarInfoView> {

    private final ICarsDataSource mCarsDataSource;

    public FullCarInfoPresenter(ICarsDataSource carsDataSource) {
        mCarsDataSource = carsDataSource;
    }

    public void acceptCar(String header, Integer id) {
```

```
        FullCarInfoView view = getView();

        addSubscription(
                mCarsDataSource
                        .acceptCar(header, id)
                        .retryWhen(new RxRetryWithDelay())
                        .subscribeOn(Schedulers.io())
                        .observeOn(AndroidSchedulers.mainThread())
                        .subscribe((t) -> view.onAccepted(), new RxErrorAction(view)));
    }

    public void removeCar(String header, Integer id) {
        FullCarInfoView view = getView();

        addSubscription(
                mCarsDataSource
                        .deleteCar(header, id)
                        .retryWhen(new RxRetryWithDelay())
                        .subscribeOn(Schedulers.io())
                        .observeOn(AndroidSchedulers.mainThread())
                        .subscribe((t) -> view.onRemoved(), new RxErrorAction(view)));
    }
}
public class LogInPresenter extends BasePresenter<LogInView> {

    private final IAccountDataSource mAccountDataSource;

    public LogInPresenter(IAccountDataSource accountDataSource) {
        mAccountDataSource = accountDataSource;
    }

    public void resendValidation(EmailDto emailDto) {
        LogInView logInView = getView();

        addSubscription(
                mAccountDataSource
                        .requestActivation(emailDto)
                        .retryWhen(new RxRetryWithDelay())
                        .subscribeOn(Schedulers.io())
                        .observeOn(AndroidSchedulers.mainThread())
                        .subscribe(
                                messageDto -> System.out.println(messageDto.getMessage()),
                                new RxErrorAction(logInView)));
    }

    public void logIn(LoginRequestDto loginRequestDto) {
        LogInView logInView = getView();

        addSubscription(
                mAccountDataSource
                        .logIn(loginRequestDto)
                        .retryWhen(new RxRetryWithDelay())
                        .subscribeOn(Schedulers.io())
                        .observeOn(AndroidSchedulers.mainThread())
                        .subscribe(logInView::processLoggingIn, new
RxErrorAction(logInView)));
    }
}
public class SignUpPresenter extends BasePresenter<SignUpView> {

    private final IAccountDataSource mAccountDataSource;

    public SignUpPresenter(IAccountDataSource accountDataSource) {
        mAccountDataSource = accountDataSource;
    }

    public void signUp(CreateAccountDto createAccountDto) {
```

```
        SignUpView signUpView = getView();

        addSubscription(
                mAccountDataSource
                        .createAccount(createAccountDto)
                        .retryWhen(new RxRetryWithDelay())
                        .subscribeOn(Schedulers.io())
                        .observeOn(AndroidSchedulers.mainThread())
                        .subscribe(signUpView::startApplication, new
RxErrorAction(signUpView)));
    }
}

public class BaseActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.fragment_container_activity);
    }

    @Override
    public void onBackPressed() {
        hideKeyboard();
        super.onBackPressed();
    }

    public void hideKeyboard() {
        View view = this.getCurrentFocus();
        if (view != null) {
            InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
            imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
        }
    }

    public void startFragment(Fragment fragment, boolean addToBackStack) {
        startFragment(fragment, addToBackStack, null);
    }

    public void startFragment(Fragment fragment, boolean addToBackStack, int animation[]) {
        hideKeyboard();

        String tag = fragment.getClass().getSimpleName();
        FragmentTransaction fragmentTransaction =
getSupportFragmentManager().beginTransaction();
        if (animation != null)
            fragmentTransaction.setCustomAnimations(
                    animation[0], animation[1], animation[2], animation[3]);
        fragmentTransaction
                .replace(R.id.container, fragment, tag)
                .setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
        if (addToBackStack) {
            fragmentTransaction.addToBackStack(tag);
        }
        fragmentTransaction.commit();
    }

    public AppComponent getAppComponent() {
        return ((App) getApplication()).appComponent();
    }
}
public abstract class BaseMvpActivity<T extends BaseView, P extends BasePresenter<T>>
        extends MvpActivity<T, P> {

    @Override
    public void onBackPressed() {
```

```
        hideKeyboard();
        super.onBackPressed();
    }

    public void hideKeyboard() {
        View view = this.getCurrentFocus();
        if (view != null) {
            InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
            imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
        }
    }

    public void startFragment(Fragment fragment, boolean addToBackStack) {
        startFragment(fragment, addToBackStack, null);
    }

    public void startFragment(Fragment fragment, boolean addToBackStack, int animation[]) {
        String tag = fragment.getClass().getSimpleName();
        FragmentTransaction fragmentTransaction =
getSupportFragmentManager().beginTransaction();
        if (animation != null)
            fragmentTransaction.setCustomAnimations(
                    animation[0], animation[1], animation[2], animation[3]);
        fragmentTransaction
                .replace(R.id.container, fragment, tag)
                .setTransition(FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
        if (addToBackStack) {
            fragmentTransaction.addToBackStack(tag);
        }
        fragmentTransaction.commit();
    }

    public AppComponent getAppComponent() {
        return ((App) getApplication()).appComponent();
    }
}
public class AddCarActivity extends BaseMvpActivity<AddCarView, AddCarPresenter> implements
AddCarView {

    @Inject
    AddCarPresenter addCarPresenter;
    @Inject
    IUserDataSource mUserDataSource;

    @BindView(R.id.description)
    EditText description;

    @BindView(R.id.name)
    EditText name;

    @BindView(R.id.price)
    EditText price;

    @BindView(R.id.year)
    EditText year;

    @BindView(R.id.image)
    ImageView mPhoto;

    public static final int REQUEST_CODE_PICK_GALLERY = 1;
    public static final int REQUEST_CODE_PICK_CAMERA = 2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_car);
```

```
    ButterKnife.bind(this);

    String image =
            savedInstanceState != null && savedInstanceState.getString("image") != null
                    ? savedInstanceState.getString("image")
                    : null;
    if (image != null) mAvatarImage = new File(image);

    String camera =
            savedInstanceState != null && savedInstanceState.getString("camera") != null
                    ? savedInstanceState.getString("camera")
                    : null;
    if (camera != null) mCameraImage = new File(camera);
}

@OnClick(R.id.image)
public void addImage() {
    pickAvatar();
}

@OnClick(R.id.get_pendings)
public void addCar() {
    if (name.getText().length() == 0) {
        Toast.makeText(this, "Fill name", Toast.LENGTH_LONG).show();
        return;
    }
    if (year.getText().length() == 0) {
        Toast.makeText(this, "Fill year", Toast.LENGTH_LONG).show();
        return;
    }
    if (price.getText().length() == 0) {
        Toast.makeText(this, "Fill price", Toast.LENGTH_LONG).show();
        return;
    }
    addCarPresenter.addCar("Bearer " + mUserDataSource.getToken(),
            CreateCarDto.builder()
                    .setDescription(description.getText().toString())
                    .setName(name.getText().toString())
                    .setYear(Integer.parseInt(year.getText().toString()))
                    .setPrice(Integer.parseInt(price.getText().toString()))
                    .build());
}

@NonNull
@Override
public AddCarPresenter createPresenter() {
    DaggerPresentersComponent.builder()
            .appComponent(getAppComponent())
            .presentersModule(new PresentersModule())
            .build()
            .inject(this);

    return addCarPresenter;
}

@Override
public Context getContext() {
    return getApplicationContext();
}

@Override
public void showError(String error) {

}

@Override
```

```
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        mAvatarImage = null;

        if (requestCode == REQUEST_CODE_PICK_GALLERY) {
            if (resultCode == Activity.RESULT_OK && preparePickedImage(data.getData())) {
                RxPickImage.sSingleton.picked(getAvatarPath());
            } else {
                RxPickImage.sSingleton.destroy();
            }
        }
        if (requestCode == REQUEST_CODE_PICK_CAMERA) {
            if (resultCode == Activity.RESULT_OK
                    && prepareCapturedImage((Bitmap) data.getExtras().get("data"))) {
                RxPickImage.sSingleton.picked(getAvatarPath());
            } else {
                RxPickImage.sSingleton.destroy();
            }
        }
    }

    private void pickAvatar() {
        if (getPackageManager().hasSystemFeature(PackageManager.FEATURE_CAMERA)) {
            PhotoGettingChoosingDialogFragment bottomSheetDialogFragment =
                    new PhotoGettingChoosingDialogFragment();
            bottomSheetDialogFragment.show(
                    getSupportFragmentManager(), bottomSheetDialogFragment.getTag());
            bottomSheetDialogFragment.setSelectListener(
                    source -> {
                        switch (source) {
                            case REQUEST_CODE_PICK_GALLERY:
                                Intent pickFromGalleryIntent =
                                        new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                                startActivityForResult(pickFromGalleryIntent,
REQUEST_CODE_PICK_GALLERY);
                                break;

                            case REQUEST_CODE_PICK_CAMERA:
//                                if (ContextCompat.checkSelfPermission(this,
Manifest.permission.CAMERA)
//                                        != PackageManager.PERMISSION_GRANTED) {
//                                    requestPermissions(new
String[]{Manifest.permission.CAMERA}, Constants.CAMERA_REQUEST_CODE);
//                                } else {
                                startCamera();
//                                }
                                break;

                            default:
                                RxPickImage.sSingleton.destroy();
                        }
                    });
        } else {
            Intent pickFromGalleryIntent =
                    new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
            startActivityForResult(pickFromGalleryIntent, REQUEST_CODE_PICK_GALLERY);
        }

        RxPickImage.getInstance(getContext(), mPhoto).pick().subscribe();
    }

    private void startCamera() {
        Intent pickFromCameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        startActivityForResult(pickFromCameraIntent, REQUEST_CODE_PICK_CAMERA);
```

```
    }

    private boolean prepareCapturedImage(Bitmap bitmap) {
        FileOutputStream fileOutputStream = null;
        try {
            fileOutputStream = new FileOutputStream(getCameraFile());
            bitmap.compress(Bitmap.CompressFormat.PNG, 80, fileOutputStream);
            return preparePickedImage(Uri.fromFile(getCameraFile()));
        } catch (Exception e) {
            return false;
        } finally {
            closeStream(fileOutputStream);
        }
    }

    private boolean preparePickedImage(Uri path) {
        InputStream inputStream = null;
        FileOutputStream fileOutputStream = null;

        boolean success = true;

        try {
            inputStream = getContentResolver().openInputStream(path);
            fileOutputStream = new FileOutputStream(getAvatarFile());

            success = inputStream != null && IOUtils.copy(inputStream, fileOutputStream) > 0;
            success = scaleImage(getAvatarFile(), 100, 100);

        } catch (Exception ex) {
            ex.printStackTrace();
            success = false;
        } finally {
            closeStream(inputStream);
            closeStream(fileOutputStream);
        }

        return success;
    }

    @Override
    public void onAdded(CarDto carDto) {
        String avatarPath = getAvatarPath();
        if (avatarPath != null) {
            addCarPresenter.uploadCar("Bearer " + mUserDataSource.getToken(), carDto.getId(),
avatarPath);
        }
        Intent intent = new Intent(this, FilterActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(intent);
    }

    public static class PhotoGettingChoosingDialogFragment extends BottomSheetDialogFragment {

        private OnSourceSelectListener mSelectListener;

        private BottomSheetBehavior.BottomSheetCallback mBottomSheetBehaviorCallback =
                new BottomSheetBehavior.BottomSheetCallback() {

                    @Override
                    public void onStateChanged(@NonNull View bottomSheet, int newState) {
                        if (newState == BottomSheetBehavior.STATE_HIDDEN) {
                            dismiss();
                        }
                    }

                    @Override
                    public void onSlide(@NonNull View bottomSheet, float slideOffset) {
```

```
                }
            };

        @Override
        public void setupDialog(Dialog dialog, int style) {
            super.setupDialog(dialog, style);
            View contentView = View.inflate(getContext(), R.layout.on_camera_clicked_layout,
null);
            dialog.setContentView(contentView);

//            View bottomSheet = dialog.findViewById(R.id.design_bottom_sheet);
//            bottomSheet.getLayoutParams().height = ViewGroup.LayoutParams.MATCH_PARENT;

            contentView.findViewById(R.id.cancel).setOnClickListener(v -> dismiss());
            contentView
                    .findViewById(R.id.choose_photo)
                    .setOnClickListener(v -> select(REQUEST_CODE_PICK_GALLERY));
            contentView
                    .findViewById(R.id.take_photo)
                    .setOnClickListener(v -> select(REQUEST_CODE_PICK_CAMERA));

            CoordinatorLayout.LayoutParams params =
                    (CoordinatorLayout.LayoutParams) ((View)
contentView.getParent()).getLayoutParams();
            CoordinatorLayout.Behavior behavior = params.getBehavior();

            if (behavior != null && behavior instanceof BottomSheetBehavior) {
                ((BottomSheetBehavior)
behavior).setBottomSheetCallback(mBottomSheetBehaviorCallback);
            }
        }

        private void select(int value) {
            mSelectListener.onSourceSelected(value);
            dismiss();
        }

        public void setSelectListener(OnSourceSelectListener selectListener) {
            mSelectListener = selectListener;
        }
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putString("image", getAvatarFile().getAbsolutePath());
        outState.putString("camera", getAvatarFile().getAbsolutePath());
    }

//    @Override
//        public void onActivityCreated(Bundle savedInstanceState) {
//        super.onActivityCreated(savedInstanceState);
//
//        String image =
//                savedInstanceState != null && savedInstanceState.getString("image") != null
//                        ? savedInstanceState.getString("image")
//                        : null;
//        if (image != null) mAvatarImage = new File(image);
//
//        String camera =
//                savedInstanceState != null && savedInstanceState.getString("camera") != null
//                        ? savedInstanceState.getString("camera")
//                        : null;
//        if (camera != null) mCameraImage = new File(camera);
//    }

    private File mAvatarImage;
```

```
    private File getAvatarFile() {
        if (mAvatarImage == null) {
            try {
                mAvatarImage = File.createTempFile("avatar", ".png",
FileHelper.cacheFolder(getContext()));
                mAvatarImage.createNewFile();
            } catch (Exception ignore) {
            }
        }

        return mAvatarImage;
    }

    private File mCameraImage;

    private File getCameraFile() {
        if (mCameraImage == null) {
            try {
                mCameraImage = File.createTempFile("camera", ".png",
FileHelper.cacheFolder(getContext()));
                mCameraImage.createNewFile();
            } catch (Exception ignore) {
            }
        }
        return mCameraImage;
    }

    private String getAvatarPath() {
        return Uri.fromFile(getAvatarFile()).toString();
    }

    private static boolean scaleImage(File originalFile, float width, float height)
            throws IOException {
        FileInputStream fileInputStream = null;
        File tmpFile =
                File.createTempFile(
                        "avatar", null, new
File(FilenameUtils.getFullPath(originalFile.getAbsolutePath())));
        try {
            fileInputStream = new FileInputStream(originalFile);
            BitmapFactory.Options options = new BitmapFactory.Options();
            options.inJustDecodeBounds = true;
            BitmapFactory.decodeStream(fileInputStream, null, options);

            closeStream(fileInputStream);

            float scale =
                    1.f / Math.max(width / (float) options.outWidth, height / (float)
options.outHeight);
            scale = Math.round(scale);
            scale = (int) Math.max(scale, 1.f);

            BitmapFactory.Options o2 = new BitmapFactory.Options();
            o2.inSampleSize = (int) scale;

            Bitmap bitmap = BitmapFactory.decodeFile(originalFile.getAbsolutePath(), o2);
            FileOutputStream fileOutputStream = new FileOutputStream(tmpFile);

            bitmap.compress(Bitmap.CompressFormat.PNG, 80, fileOutputStream);

            closeStream(fileOutputStream);
            bitmap.recycle();

            IOUtils.copy(new FileInputStream(tmpFile), new FileOutputStream(originalFile));

            if (!tmpFile.delete()) tmpFile.deleteOnExit();
```

```
        } finally {
            closeStream(fileInputStream);
        }
        return true;
    }

    private static void closeStream(@Nullable Closeable closeable) {
        if (closeable != null) {
            try {
                closeable.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    public interface OnSourceSelectListener {

        void onSourceSelected(int source);
    }
}

public class CarListActivity extends BaseActivity {

    @BindView(R.id.toolbar)
    Toolbar toolbar;
    private Drawer.Result drawerResult;

    @Inject
    IUserDataSource mUserDataSource;

    @BindView(R.id.list)
    RecyclerView list;

    private CarsAdapter carsAdapter;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_car_list);

        ButterKnife.bind(this);

        DaggerPresentersComponent.builder()
                .appComponent(getAppComponent())
                .presentersModule(new PresentersModule())
                .build()
                .inject(this);

        List<CarDto> carDtoList = (ArrayList<CarDto>)
getIntent().getSerializableExtra(Constants.CAR_LIST);
        RecyclerView.LayoutManager layoutManager
                = new LinearLayoutManager(getApplicationContext(),
LinearLayoutManager.VERTICAL, false);
        list.setLayoutManager(layoutManager);
        carsAdapter = new CarsAdapter();
        if (carDtoList != null) {
            carsAdapter.add(carDtoList);
        }
        list.setAdapter(carsAdapter);

        initializeToolbar();
        initializeNavigationDrawer();
    }

    @Override
```

```
    public void onBackPressed() {
        if (drawerResult.isDrawerOpen()) {
            drawerResult.closeDrawer();
        } else if (getFragmentManager().getBackStackEntryCount() != 0) {
            getFragmentManager().popBackStackImmediate();
        } else {
            super.onBackPressed();
        }
    }

    private void initializeToolbar() {
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowTitleEnabled(false);
    }

    private void initializeNavigationDrawer() {
        drawerResult = new Drawer()
                .withActivity(CarListActivity.this)
                .withToolbar(toolbar)
                .withActionBarDrawerToggle(true)
                .addDrawerItems(
                        new PrimaryDrawerItem().withName(getString(R.string.log_out))
                                .withIdentifier(1),
                        new PrimaryDrawerItem().withName(getString(R.string.add_add))
                                .withIdentifier(2)
                )
                .withOnDrawerListener(new Drawer.OnDrawerListener() {
                    @Override
                    public void onDrawerOpened(View drawerView) {
                        InputMethodManager inputMethodManager
                                = (InputMethodManager) getSystemService(
                                Activity.INPUT_METHOD_SERVICE);
                        inputMethodManager
                                .hideSoftInputFromWindow(getCurrentFocus()
                                        .getWindowToken(), 0);
                    }

                    @Override
                    public void onDrawerClosed(View drawerView) {
                    }
                })
                .withOnDrawerItemClickListener((parent, view, position, id, drawerItem) -> {
                    int drawerItemId = drawerItem.getIdentifier();
                    switch (drawerItemId) {
                        case 1:
                            mUserDataSource.clear();
                            Intent intent = new Intent(this, WelcomeActivity.class);
                            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                            startActivity(intent);
                            break;
                        case 2:
                            Intent intent1 = new Intent(getBaseContext(),
AddCarActivity.class);
                            startActivity(intent1);
                            break;
                    }
                })
                .build();
    }

    class CarsAdapter extends SimpleAdapter<CarDto, RecyclerView.ViewHolder> {

        @Override
        public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
            return new
```

```
ViewHolder(LayoutInflater.from(parent.getContext()).inflate(R.layout.car_item_layout, parent,
false));
        }

        @Override
        public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
            if (getItem(position).getImageUrl() != null) {
                OkHttpClient client = new OkHttpClient.Builder()
                        .addInterceptor(chain -> {
                            Request newRequest = chain.request().newBuilder()
                                    .addHeader("Authorization", "Bearer " +
mUserDataSource.getToken())
                                    .build();
                            return chain.proceed(newRequest);
                        })
                        .build();

                Picasso picasso = new Picasso.Builder(getApplicationContext()).downloader(new
OkHttp3Downloader(client)).build();

                picasso.load(getItem(position).getImageUrl())
                        .into(((ViewHolder) holder).image);
            } else {
                Picasso.with(getApplicationContext())
                        .load(R.drawable.car_default)
                        .into(((ViewHolder) holder).image);
            }

            ((ViewHolder) holder).name.setText(getItem(position).getName());
            ((ViewHolder) holder).year.setText(String.valueOf(getItem(position).getYear()));
            ((ViewHolder) holder).price.setText(String.valueOf(getItem(position).getPrice()));
        }

        class ViewHolder extends RecyclerView.ViewHolder {

            @BindView(R.id.image)
            ImageView image;
            @BindView(R.id.name)
            FontTextView name;
            @BindView(R.id.year)
            FontTextView year;
            @BindView(R.id.price)
            FontTextView price;

            public ViewHolder(View itemView) {
                super(itemView);
                ButterKnife.bind(this, itemView);
            }

            @OnClick
            public void onClick(final View view) {
                int itemPosition = list.getChildLayoutPosition(view);
                Intent intent = new Intent(getBaseContext(), FullCarInfoActivity.class);
                intent.putExtra(Constants.CAR, getItem(itemPosition));
                startActivity(intent);
            }
        }
    }
}
public class FullCarInfoActivity extends BaseMvpActivity<FullCarInfoView,
FullCarInfoPresenter> implements FullCarInfoView {

    @Inject
    IUserDataSource mUserDataSource;
    @Inject
    FullCarInfoPresenter fullCarInfoPresenter;
```

```
@BindView(R.id.image)
ImageView image;
@BindView(R.id.name)
TextView name;
@BindView(R.id.year)
TextView year;
@BindView(R.id.price)
TextView price;
@BindView(R.id.description)
TextView description;
@BindView(R.id.accept)
View acceptButton;
@BindView(R.id.remove)
View removeButton;
@BindView(R.id.write)
View writeButton;

private CarDto carDto;

@BindView(R.id.toolbar)
Toolbar toolbar;
private Drawer.Result drawerResult;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_full_car_info);

    ButterKnife.bind(this);

    carDto = (CarDto) getIntent().getSerializableExtra(Constants.CAR);
    initialize(carDto);

    if (mUserDataSource.getRole().equals(AccountRole.ADMINISTRATOR.toString())) {
        if (!carDto.getEnabled()) {
            acceptButton.setVisibility(View.VISIBLE);
        }
        removeButton.setVisibility(View.VISIBLE);
    }
    if (mUserDataSource.getUserLogin().equals(carDto.getLogin())) {
        removeButton.setVisibility(View.VISIBLE);
        writeButton.setVisibility(View.INVISIBLE);
    }

    initializeToolbar();
    initializeNavigationDrawer();
}

@Override
public void onBackPressed() {
    if (drawerResult.isDrawerOpen()) {
        drawerResult.closeDrawer();
    } else if (getFragmentManager().getBackStackEntryCount() != 0) {
        getFragmentManager().popBackStackImmediate();
    } else {
        super.onBackPressed();
    }
}

@OnClick(R.id.write)
public void writeToOwner() {
    Intent i = new Intent(Intent.ACTION_SEND);
    i.setType("message/rfc822");
    i.putExtra(Intent.EXTRA_EMAIL, new String[]{carDto.getLogin()});
    i.putExtra(Intent.EXTRA_SUBJECT, "I want to buy a car");
    try {
        startActivity(Intent.createChooser(i, "Send mail..."));
```

```
        } catch (android.content.ActivityNotFoundException ex) {
            Toast.makeText(getApplicationContext(), "There are no email clients installed.",
Toast.LENGTH_SHORT).show();
        }
    }

    @OnClick(R.id.remove)
    public void removeCar() {
        fullCarInfoPresenter.removeCar("Bearer " + mUserDataSource.getToken(),
carDto.getId());
    }

    @OnClick(R.id.accept)
    public void acceptCat() {
        fullCarInfoPresenter.acceptCar("Bearer " + mUserDataSource.getToken(),
carDto.getId());
    }

    private void initialize(CarDto carDto) {
        if (carDto.getImageUrl() != null) {
            OkHttpClient client = new OkHttpClient.Builder()
                    .addInterceptor(chain -> {
                        Request newRequest = chain.request().newBuilder()
                                .addHeader("Authorization", "Bearer " +
mUserDataSource.getToken())
                                .build();
                        return chain.proceed(newRequest);
                    })
                    .build();

            Picasso picasso = new Picasso.Builder(getApplicationContext()).downloader(new
OkHttp3Downloader(client)).build();

            picasso.load(carDto.getImageUrl())
                    .into(image);
        } else {
            Picasso.with(getApplicationContext())
                    .load(R.drawable.car_default)
                    .into(image);
        }

        name.setText(carDto.getName());
        year.setText(String.valueOf(carDto.getYear()));
        price.setText(String.valueOf(carDto.getPrice()));
        description.setText(String.valueOf(carDto.getDescription()));
    }

    @NonNull
    @Override
    public FullCarInfoPresenter createPresenter() {
        DaggerPresentersComponent.builder()
                .appComponent(getAppComponent())
                .presentersModule(new PresentersModule())
                .build()
                .inject(this);

        return fullCarInfoPresenter;
    }

    @Override
    public void onRemoved() {
        Toast.makeText(getContext(), "Removed", Toast.LENGTH_LONG).show();
    }

    @Override
    public void onAccepted() {
        Toast.makeText(getContext(), "Accepted", Toast.LENGTH_LONG).show();
```

```
    }

    @Override
    public Context getContext() {
        return getApplicationContext();
    }

    @Override
    public void showError(String error) {
        Toast.makeText(getContext(), error, Toast.LENGTH_LONG).show();
    }

    private void initializeToolbar() {
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowTitleEnabled(false);
    }

    private void initializeNavigationDrawer() {
        drawerResult = new Drawer()
                .withActivity(FullCarInfoActivity.this)
                .withToolbar(toolbar)
                .withActionBarDrawerToggle(true)
                .addDrawerItems(
                        new PrimaryDrawerItem().withName(getString(R.string.log_out))
                                .withIdentifier(1),
                        new PrimaryDrawerItem().withName(getString(R.string.add_add))
                                .withIdentifier(2)
                )
                .withOnDrawerListener(new Drawer.OnDrawerListener() {
                    @Override
                    public void onDrawerOpened(View drawerView) {
                        InputMethodManager inputMethodManager
                                = (InputMethodManager) getSystemService(
                                Activity.INPUT_METHOD_SERVICE);
                        inputMethodManager
                                .hideSoftInputFromWindow(getCurrentFocus()
                                        .getWindowToken(), 0);
                    }

                    @Override
                    public void onDrawerClosed(View drawerView) {
                    }
                })
                .withOnDrawerItemClickListener((parent, view, position, id, drawerItem) -> {
                    int drawerItemId = drawerItem.getIdentifier();
                    switch (drawerItemId) {
                        case 1:
                            mUserDataSource.clear();
                            Intent intent = new Intent(this, WelcomeActivity.class);
                            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                            startActivity(intent);
                            break;
                        case 2:
                            Intent intent1 = new Intent(getBaseContext(),
AddCarActivity.class);
                            startActivity(intent1);
                            break;
                    }
                })
                .build();
    }
}
```